BEBR
FACULTY WORKING
PAPER NO. 90-1624

Incorporating Machine Learning in Knowledge-Based Process Planning Systems: An Explanation-Based Approach

Michael J. Shaw
Sangchan Park
Uday Menon

College of Commerce and Business Administration
Bureau of Economic and Business Research
University of Illinois Urbana-Champaign

FACULTY WORKING PAPER NO. 90-1624

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

January 1990

Incorporating Machine Learning in Knowledge-Based Process Planning Systems: An Explanation-Based Approach

Michael J. Shaw

Sangchan Park

Uday Menon

The Beckman Institute for Advanced Science and Technology

University of Illinois at Urbana-Champaign

ABSTRACT

This paper presents a new approach using machine learning to automate process planning. Unlike most of the existing AI-based process planning systems, our system, referred to as LASIPP (Learning Augmented System for Intelligent Process Planning), enables the process planning system to refine its knowledge or acquire new knowledge by deriving and observing example plans. Specifically, the explanation based learning (EBL) technique is used in LASIPP for knowledge acquisition and refinement in the process planning domain. We will show that the EBL capability enables the process planning system to learn new heuristics or schemata for achieving better performance. As such, this learning augmented approach is shown to provide an appealing framework for integrating the generative and the variant methods for automatic process planning.

1 INTRODUCTION

Modern manufacturing technology's solution to improving productivity has resulted in Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) systems. The gains obtaining from CAD and CAM alone are not, however, sufficient to meet the anticipated performance, and the integration of CAD/CAM appears to hold the key to further productivity gains. This integration would require automating the process planning function which serves as an interface between CAD and CAM [Chang & Wysk, 1984]. There has been a great deal of research effort applying artificial intelligence (AI) methods to automate process planning, mostly making use of knowledge-based systems to generate process plans.

This paper proposes augmenting an automated process planning system by fortifying it with machine learning capabilities. A system capable of learning from its experience in creating process plans and observing plans created by others can, like a human process planner, improve its performance over time. This improvement is demonstrated in two distinct ways: the ability to create better plans from an optimality standpoint, and a reduction in time required to create these plans. A learning augmented system develops the ability to recognize general patterns in sequences of machining operations and indexes these patterns by the features created by them, much like a human expert indexing solutions by observable characteristics of the problems they help solve [Larkin et al., 1980]. By being able to "see" a complex problem (in this case, the generation of a process plan that will create the features specified by the part design) as a

combination of a few subproblem chunks (general patterns), each of which can be solved by a known sequence of machining operations, the solution to the complex problem is expedited. This is because each subproblem chunk is solved in "one sweep" by this known sequence. A useful analogy may be to imagine solving a large jigsaw puzzle consisting of 2,000 pieces. If one knew how three (nonintersecting) subsets of 500, 500 and 1,000 pieces fit together, the puzzle is nowhere near as intimidating as if one had to put it together one piece at a time. We employ the Explanation Based Learning (EBL) [DeJong, 1985] algorithm to help an automated process planning system recognize these general patterns (called schemata) in manufacturing processes. These schemata are added to the system's knowledge base as and when learning takes place.

The implications of using EBL for process planning are thus twofold:

(a) The creation and deployment of schemata are in keeping with the Group Technology principle of "taking advantage of similarity," since they eliminate the wasteful redundant effort that goes into creating process plans for parts with common features. EBL provides an intelligent method for extracting parts of a process plan which can be directly applied in the future for the creation of a similar plan.

(b) EBL is able to combine the advantages of generative planning systems (the creation of a process plan from a given set of operators) with those of variant planning systems (the savings obtaining from standardization). This unification of

two, hitherto distinct, process planning approaches has a synergistic bearing on the capability of such an automated system for process planning.

Thus, while other automated systems are equipped with a static knowledge-base that can be enlarged only by manually adding more decision rules from time to time, an EBL based (or more generally, a machine learning based) process planning system, has a dynamic knowledge-base that grows with "experience." This we believe is the key to building intelligent process planning systems. As we shall show, integration of the AI problem solving and machine learning in what we call a Learning Augmented System for Intelligent Pocess Planning (LASIPP), can overcome some of the limitations of existing process planning approaches.

The rest of the paper is organized as follows: Section 2 presents a discussion of the issues involved in automated process planning and a survey of existing automated process planning systems. Section 3 describes the EBL algorithm while the details of implementing an EBL based process planning system are laid out in Section 4. In Section 5, we show how such a system understands a given example process plan and learns new heuristics. In Section 6, the advantages of learning augmented process planning systems are brought into focus by contrasting them with existing approaches to automated process planning.

2 AUTOMATED PROCESS PLANNING

Process planning is concerned with the task of determining a sequence of machining processes, and the choice of cutting tool parameters that will transform the workpiece from the given initial state

to the desired goal state specified by the engineering design. Numerous factors, such as shape, tolerance, surface finish, size, material type, quantity, and the manufacturing resources available, affect process planning and must be taken into account.

Process planning draws heavily on empirical knowledge acquired by experience. The way this planning knowledge is organized (stored) determines its relative ease of retrieval and use in creating process plans. Thus if a prescriptive form of knowledge organization is chosen (e.g., "to make a widget XYZ, use the plan which consists of manufacturing operations 01, 02, and 05 in sequence"), the use of this knowledge is straightforward once it is retrieved. On the other hand, if the knowledge is stored in a descriptive format (e.g., "operation 02 creates a flat surface"), its use in putting together a process plan for "widget XYZ" is not quite as straightforward as before. A search must be conducted of the knowledge base for suitable manufacturing operations, represented by "operators" [Fikes et al. 1981], that will generate all the required features on our widget, and then these operators must be strung together in a viable sequence to yield the plan. While the descriptive method of organizing knowledge lends greater flexibility in putting together process plans (there is more than one way to skin a cat), this must be weighted against the increased computational resources required by this method. The class of automated planning systems that use prescriptive knowledge organization are called variant systems while those employing descriptive knowledge organization are called generative systems. Table 1 presents a comparative analysis of these systems.

Insert Table 1 about here

Planning systems have not been able to completely replace the human element and typically require human inputs to interpret problem specifications and modify standard plans. What is missing in both types of systems is the ability to understand and learn new concepts based on the system's experience in creating process plans and/or observing example plans input to the system. Satisfaction of these requirements would entail augmenting the system with a learning unit.

LASIPP integrates such a Learning Unit with the typical functional modules of an automated process planning system, including a process planning unit, and a knowledge base (see Figure 2-1). The knowledge base contains operators and heuristic rules for their application. These are used by the planner to chart out a problem solving strategy. Since the learning unit seeks to constantly refine existing rules and add new rules to the system's knowledge base, the heuristic power of the system--measured by the optimality of the solutions and the speed with which it generates them--is thus constantly improved by the Learning Unit.

Insert Figure 2-1 about here

The Learning Unit is the key feature distinguishing LASIPP from a generative system such as the SIPP system [Nau & Chang, 1986]. Given the domain knowledge (set of operators, objects and decision logic) and a planning problem, a generative system proceeds to chain together a set of operators based on its decision logic and stops when a

complete process plan has been created. By contrast, LASIPP's function does not end here. Instead, its learning unit is triggered at this point. The learning unit first understands the plan which has been created by following an inference process and building causal relationships between states and operators used. It next proceeds to run some tests on a generalized version of the plan to determine if it is worthwhile storing for future use. If so, the generalized plan is added to the system's knowledge base. This process of making generalization of an existing concept is crucial for machine learning [Mitchell, 1981]. We shall show that this same generalization process will be very useful in deriving schemata from existing process plans.

When faced with a subsequent planning problem similar to the one just solved, LASIPP simply retrieves the stored generalized plan, the schema, and instantiates it with problem specific parameters, thus saving considerable effort. In this fashion, given the same initial domain knowledge, LASIPP expands it by constantly refining its heuristic rules and adding new rules to the domain. We shall show that the learned knowledge in LASIPP help save considerable searching effort in generating process plans.

3 EXPLANATION BASED LEARNING (EBL): THE METHODOLOGY

3.1 Operational Definition & Assumptions of EBL

Recognized as the essential feature of any intelligent system, learning processes include the acquisition of new declarative knowledge, the development of problem-solving skills through instruction or practice, the organization of new knowledge into general, effective

representations, and the discovery of new facts and theories through observation and experimentation. Machine learning is concerned with the computer modeling of the learning processes. Machine learning methods can be categorized into the following areas based on their behavioral characteristics: rote learning [Samuel, 1968], learning from instruction [Davis, 1979], learning by induction [Dietterich and Michalski, 1983; Shaw, 1987], learning by analogy [Carbonell, 1983], learning by competition [Holland, 1986], and learning from observation and discovery [DeJong, 1986; Langley, 1981].

EBL [DeJong & Mooney, 1986] can be characterized as learning from observation, since the learner is not provided with a set of examples of a particular concept. Rather, it must make its own observation about the concept. As a machine learning technique, the twin advantages of learning without supervision and being able to learn from a single example characterize this learning method.

The observation can be based on either a single event or a set of events resulting from a planning process. The EBL system then attempts to generalize these events into a learned concept. First, EBL attempts to match the preconditions and the effects of successive actions (operators) and explains the application and sequence of several operators in a given plan in terms of this matching, resulting in a solution tree called the explanation structure. This solution tree is subsequently retained by the system and, through the generalization process, translated into a special format called a schema. When the system is faced with a problem that bears similarities with the one for which the schema (i.e., the stored plan) is a solution,

it simply retrieves the schema and instantiates (binds variables to values) it with specific values associated with the problem encountered.

The following are required to be given as inputs to EBL: (1) domain theory, (2) initial state, (3) goal state, and (4) an example plan (optional). Descriptions for these EBL components are in order.

(1) Domain theory. This consists of five components:

- A specification of types of resources and their properties.
- A specification of types of objects and their properties.
- Problem solving operators for representing procedural knowledge.
- A set of inference rules for inferring properties and relations, including heuristics.
- A set of schemata which are previously learned.

The first three components are common to any problem solver, such as the STRIPS system described in [Fikes et al. 1981] and the knowledge-based scheduler in [Shaw 1988]. Schemata are similar to macro-operators (canned solutions) maintained by other systems, except that they exist in an uninstantiated (see below) generalized form, while macro operators tend to be more specific since they are generated by a data driven approach.

(2) Initial state. A specification of objects in the world and their properties.

(3) Goal state. A general specification of a goal. In general, a goal is an incomplete world state, yet to be achieved.

(4) An example plan. This is a sequence of domain operators that will transform the initial state to the goal state. When a plan to achieve the goal is required, the planner module constructs this sequence by performing a search of the domain. Alternatively, this sequence may be input to the system in which case the objective is to get the learner module to understand the given plan and refine/enhance system knowledge. The characteristics of EBL enable the acquisition of knowledge (learning) from a single example plan, even if this plan is less than optimal. Thus it is not necessary that a process planning expert input the example plan as long as some feasible plan is input. Given such a feasible plan and an adequate domain theory, the EBL process is able to create an efficient generalized plan. It is important to note that such inputs of example plans serve to focus the algorithm's knowledge acquisition effort, thereby accelerating the learning process.

3.2 The Algorithmic Procedure of EBL

There are three distinct steps in the knowledge acquisition cycle of EBL. These are:

- Understanding either a given example plan or a plan constructed by the Planning Unit;
- Evaluating the plan to see if it is worthwhile internalizing; and
- Generalizing the plan to form a new schema.

(I) Understanding and explaining. Once a plan to achieve the given goal is available (as described above, this is either created by the planner module or is given externally), EBL begins by understanding

the input. Understanding involves building the explanation structure by generating the proof tree starting from the initial state and applying the operator sequence of the plan. It is necessary that the system maintain causal relationships justifying every element, including all relevant objects and their properties in the representation. When an inference rule or operator is invoked during this phase, a copy of this is added to the explanation structure. Since this copy of the rule/operator has variables as arguments, these variables must be bound to problem specific values before they can relate to the given problem. This binding of variables to a set of values is called instantiation. The variable-value bindings are specified by a specific substitution list S. This list ensures that the initial state matches the preconditions of the first operator. For subsequent operators, the list ensures that the consequents of one operator match the antecedents of the next operator in the sequence. Note that the S list is generated by the system. In the case of creating a plan to achieve a given goal, the planning module starts from the goal and uses backward chaining to find a suitable operator sequence. Every time an operator is found, it is instantiated such that the consequents of the operator match the antecedents of the next operator in the forward sequence (i.e, the operator closer to the goal state). The system keeps track of these bindings in a list S that is constantly updated. This procedure of building the S list remains the same for the case when a plan is input to the system, the only difference being that in the latter case the planner is only required to search the domain for any missing links identified in the given plan.

Once an explanation structure and an S list are constructed as described, the understanding phase (also called the explanation phase) is complete. A failure at this stage aborts the algorithm and no learning takes place.

(II) Evaluation. The explanation structure is a skeleton comprising a sequence of operators and inference rules in uninstantiated form. When this skeleton is instantiated using the S list, it takes on the form of an explanation of how this sequence transforms the given initial state into the desired goal state. In the process planning context, the explanation constitutes the required process plan. The system now evaluates this plan to see if a generalized version of it should be stored in the knowledge base for future uses. It does this using a five point checklist [DeJong, 1983] of criteria that all generalized plans must satisfy. This checklist verifies that (1) the goal is achieved; (2) the goal is a general one; (3) the operators required are available; (4) the method of achieving the goal is at least as effective as other known ones; (5) the input matches a known generalizable pattern. Note that the evaluation criteria are applied to a generalized plan. This is obtained by repeating the understanding process and building a general substitution list G, which replaces the constants of the S list with variables while maintaining the constraints on variable-value bindings as before. In fact, the G & S lists are simultaneously built and updated in one pass of the proof tree.

(III) Generalization. A generalized plan that meets the evaluation criteria undergoes some transformation before it is internalized.

Rather than store all the nodes of the explanation structure, the system only stores those that causally support the goal and can be easily recognized by it; that is, only those nodes that satisfy the operationality criterion [Mitchell et al., 1986]. Since the system must be able to retrieve a generalized plan by recognizing that a given specification of initial and goal states is an instance of the generalized plan, it ensures that the specifications of the leaf nodes (of the generalized plan) are easily recognizable; i.e., they are operational. Thus if node A is a leaf node that supports node B, and node B is operational, then node A is pruned from the explanation structure. In addition, segments of the structure that can be more efficiently achieved by known schemata are replaced by such schemata. Finally, the remaining explanation structure is instantiated using the G list to yield the generalized plan that is stored in a schema format. A schema is indexed by a set of preconditions and effects which correspond to generalizations of the initial and goal states of some previously encountered plan based on which this schema was generated.

4 THE PLANNING UNIT OF LASIPP

The present implementation of LASIPP assumes that the input to the system will be a solid model representation of objects. Based on the representation, a pre-processor will perform a "subtraction" between the goal state (the desired part) and the initial state (the raw material); the result is referred to as the total difference shape (TDS). Thus, similar to the means-ends analysis described in [Nilsson, 1980], the inference process carried out by the planning

unit of LASIPP seeks to apply rules/operator in its knowledge-base to reduce the given TDS. When TDS becomes zero, the goal state is reached and the plan is completed.

The shape of parts is recognized as particular instances of a set of generic shapes with the aid of inference rules. Figure 4-1 is the hierarchy of shape primitives in relation to their basic operations and achieved features. These shapes describe the relative notion between the cutting tool and workpiece. This is illustrated in Figure 4-2 where the shaded rectangular area represents the cross-section of the cutting tool embedded in a cylindrical workpiece. By moving this tool over the surface of the cylinder in a circular motion, a volume of material in the shape of a hollow cylinder is removed. We use the name "cylinder difference" to describe this volume of material, and classify this shape as a particular instance of a more general shape called Rotational Sweep (class of shapes obtained by all possible rotary relative motions). By the same token, a cuboid is classified as a particular instance of Translational Sweep [Chang & Wysk, 1985, p. 73], obtained by a purely linear relative motion between tool and workpiece (shown in the right half of Figure 4-2).

Insert Figures 4-1 and 4-2 Here

Each shape is associated with a property list which comprises of the minimum number of independent dimension specifications required to describe the shape. For example, a cuboid would have length, width, and thickness as its three properties, while a cylinder would be represented by length, ID (inner diameter) and OD (outer diameter).

The domain theory would include an exhaustive set of primitives. These primitive shapes will be the building blocks used to create more complex shapes.

Figure 4-3 shows a sample collection of inference rules as they would appear in domain theory. These rules have a frame-like structure with slots for rule name, argument list, the rule antecedents and the rule consequents.

Insert Figure 4-3 Here

Each machining process for material removal and surface alteration is indexed by a set of preconditions (antecedent) that must be satisfied and effects (consequent) of applying the operator. Preconditions include process capability information, shape of the difference to be removed and surface accessibility for application of that operator. For example, Figure 4-4 illustrates some sample operators including the operators representing the rough-finish milling and twist drilling operators.

Insert Figure 4-4 Here

We shall now use a process planning example to illustrate the problem-solving and learning process in LASIPP. The engineering design of the part under planning is shown in Figure 4-5. Figure 4-6 shows the corresponding initial and goal state descriptions generated by the preprocessor based on the given part design; these state descriptions will be the input to LASIPP. The nine differences D1-D9

making up the TDS, can be removed by the rough-finish-milling and twist-drilling operators. Figure 4-7 illustrates this volume removal process, where the shaded volume corresponds to the finished part. In Figure 4-6, each face of the finished part has been labelled by a three letter name where the first letter designates the face while the next two letters specify the orientation of that face w.r.t. a fixed set of coordinates XYZ. Thus label BXP, for example, is used to designate the larger shaded face B which has an orientation of X-axis Positive. Descriptions of the finished part and the rectangular block from which it is to be created are specified in Hi-Mapp's Form Feature Language [Benerji & Khoshnevis, 1986] in Figure 4-7. A viable process plan for making this part is shown in Figure 4-8.

Insert Figures 4-5, 4-6, 4-7 & 4-8 Here

The output of this stage is a partially ordered list of volume differences D. Two transformations have in effect been performed on the input: First, the input has been parsed into a goal specification that is described by the TDS; and second, the TDS has been factored into a set of subdifferences that can be removed using domain operators, and a nonlinear plan for the removal of these subdifferences has been created. The result of the planning process is a process plan, as the one shown in Figure 8, which is a plan of machining activities.

To demonstrate the learning capability of LASIPP, we will use the machining of the combination of D8 (a cuboid difference) and D9 (a cylinder difference) to demonstrate the schema acquisition capability of LASIPP.

5 KNOWLEDGE ACQUISITION, REFINEMENT, AND APPLICATION

We motivate the discussion in this section with a recapitulation of the key features of LASIPP, showing how they tie in with our primary objective of generating process plans efficiently and quickly. Given a planning problem, we require our system to recognize familiar patterns in the problem's requirements so that it can partition the problem into subproblem chunks, solving each with an existing schema. Having created a process plan for the planning problem in this way, our system must also internalize any new patterns that it sees as worthy of storage for future use. EBL, as described in Section 3, helps LASIPP recognize new patterns and internalize them. The Understanding step of the algorithm finds these two patterns by looking at the structure of the solution (sequence of operators) and explaining why this structure is a legitimate (feasible) one. The Evaluation and Generalization steps are concerned with the process of internalizing these patterns, indexing them by recognizable (this is the operationality criterion described in Section 3) problem characteristics. We will illustrate the mechanics of EBL with the help of the aforementioned example illustrated in Figures 4-5 to 4-8.

Without loss of generality, let us focus on a part of this process planning problem--the removal of differences D8 and D9--in order to see how EBL functions. Figure 5-1 describes the learning problem for this example. As shown, the problem is viewed as one of learning the concept of "block-with-hole." The resulting schema represents a new piece of knowledge that can be used in subsequent process planning. A

step-by-step description of how the system creates this plan is given as follows.

Insert Figure 5-1 Here

(I) Building the Explanation

As described in Section 3.2, the first stage in the EBL process is building an explanation about how the part can be produced. LASIPP does this by performing backward chaining starting from the goal. It is important to note that the building of an explanation structure and the construction of a process plan can be simultaneously achieved.

The goal described in Figure 5-1 consists of two subgoals: a shoulder (flat surface) and a hole (straight-hole). The machining processes corresponding to these two subgoals are illustrated in Figure 5-2. Starting with the latter, the system searches for an operator or an inference rule that has "straight-hole (?od ?length tol ?h)" as a consequent, which is inference rule I3 (see unification U1 in Figure 5-3).

Insert Figures 5-2 and 5-3 Here

As the subgoal "straight-hole" is satisfied the consequents of the inference rule I3, the antecedent of the rule I3 "cylinder removed (?id3, ?od3, ?length3, ?tol3, ?h3)" becomes a new subgoal. On finding that the consequent of the operator "twist drilling" provides a possible match with this new subgoal (two of its three preconditions are met by the initial state), an uninstantiated version of this operator

is retrieved from domain theory (thus building the first part of the explanation structure), and the specific (S) and general (G) substitution lists are updated by adding the necessary variable bindings required to unify the consequents of this operator with the goal (see Figure 5-3 and 5-4 Unification U2).

S & G are updated again when the preconditions (or antecedents) "placed" and "cylinder difference" are unified with the initial state (see U3 and U5 in Figure 5-3 and 5-4). Since the third precondition "flat surface" is yet to be satisfied, the backward chaining process continues. As before, "flat surface" is parsed into "tsweep removed (?length2, ?lwidth2, ?thick2, ?tol2, ?fs2)" by inference rule I2, and a search is conducted for an operator whose consequent matches this new subgoal. This results in the selection of the rough-finish milling operator as a possible candidate. One precondition of rough-finish milling is met by the initial state "cuboid difference" so unification is performed, yielding new S and G lists.

Insert Figure 5-4 about here

Note that the system must first infer that "cuboid difference" is a special case of "tsweep difference" before it can verify this match. LASIPP achieves this inference of the generalization relation by checking the hierarchy shown in Figure 4-1. The other precondition which remains to be verified is a set of boundary conditions specifying process capability. This is verified to be true for the given planning problem, thus completing the construction of a process plan and explanation structure.

(II) Evaluation

Instantiating the explanation structure using the G list yields a generalized explanation in which all piecewise rules and operators are indexed (linked) together to explain the process of making the part. This generalized plan for achieving the block-with-hole goal is now put to the evaluation test where questions such as goal generality, plan optimality and resource availability must be answered. For example, one measure of goal generality could be the number of times this goal is encountered by the system.

(III) Generalization

Once the generalized explanation passes the evaluation test, a schema is created. Since, based on the domain hierarchy in Figure 4-1, the state "cuboid difference" is a special case of the state "tsweep difference," which meets one of the preconditions of the rough-finish milling operator, this state ("cuboid difference") is pruned from the explanation. This is an instance of the application of the operationality criterion described in Section 3.2. The general explanation that remains after this single excision must be packaged into a schema. The leaf nodes of the explanation ("tsweep difference," "operator boundary conditions" and "placed") become the antecedents of the schema which has a rule-like structure. Similarly, "flat-surface" and "straight-hole" become the consequents. In addition, the schema must also include the operator sequence that constitutes the generalized process plan. Figure 5-5 shows the schema for this example problem.

Insert Figure 5-5 about here

Inference rule I2, as described in Figure 4-3, is necessary to link the state "tsweep removed (?length2, ?width2, ?thick2, ?tol2, ?fs2)," which is operational, with the state "flat surface (?length2, ?width2, ?thick2, ?tol2, ?fs2)," which is a more abstract specification of the subgoal in nonoperational terms. In the absence of this rule, LASIPP will not be recognized that the subgoal (of creating a flat surface) has been achieved, and will continue to search for other (nonexistent) operators that can reduce this perceived difference. Thus inference rules such as I2 perform an important state recognition function. Their inclusion in the explanation structure slot of a schema saves the searching time in future process plans. Inference rules in effect act as an interface between high level feature specifications such as "straight-hole" (which is used in the solid model), and low level feature specifications that are expressed in operational terms which the system can recognize. This transformation is an important function of EBL [Keller, 1988].

6 AN EXAMPLE USING LASSIP TO GENERATE THE PLAN FOR MAKING AXLE COVERS

Having demonstrated how LASIPP creates a process plan and acquires a new schema in the context of the block-with-hole example, we now turn to an examination of using LASIPP for a more complicated process planning problem. Figure 5-6 shows the engineering part design for an Axle Cover--a component used in automotive transmissions. The features of this part include a spigot, four threaded blind holes, two threaded through holes, a center bore and several others. The creation of this part from the raw casting includes the removal of 16 volume differences

in seven different machine setups. Using LASIPP's previously described methodology, the initial and goal state descriptions together with a listing of the volumes differences are laid out in Figures 5-7 to 5-9. By chaining domain operators together in a particular sequence as dictated by the constraints of the problem, LASIPP creates a process plan for the axle cover as shown in Figure 5-10.

Insert Figures 5-6 to 5-10 Here

The learning unit takes this plan as input and produces a generalized plan from it. As described in Section 3, this is achieved by replacing each argument of the goal-state predicate with a distinct variable and propagating these back through the explanation structure to determine the appropriate unifications that must be preserved for this plan to succeed.

Having obtained a generalized plan for the axle cover in this manner, the next step is schema acquisition. Should the entire generalized plan for the axle cover be packaged as a schema or should the system focus on only a part of this plan? Choice of the latter alternative gives rise to a fresh concern. Which part of the plan should the system schematize?

Clearly, the answers to these questions must be provided by the features of the part created. The abstraction of parts of a process plan must be based on the degree of similarity that the associated group of features (created by this abstracted plan) bears to other frequently encountered feature groups. What we are in fact suggesting

is that the system acquire schemata which can be used to create groups of features that are common to several part designs.

Note the close parallel between the idea developed for group technology that one standard plan should be used for all members of a part family, and our notion of one schema for all members containing the same feature group. Thus, while variant planning systems use commonalities in shape as a criterion for part family classification, we advocate using commonalities in features (such as center-hole-with-chamfer-currounded-by-four-threaded-holes) as the criterion for feature family classification. The underlying logic for doing this is that engineering design is function-directed by nature, so two parts A & B that have similar function descriptions will generally share a common subset of features as well. If A & B differ in overall shape, they will belong to different part families (implying that a separate standard plan exists for each), and a variant system is unable to take advantage of the commonalities in features between them. LASIPP on the other hand, is able to exploit this commonality by using one schema to create this subset of features in both A & B. More importantly, EBL provides LASIPP with the ability to make generalization of the schema. That is, the system is capable of deriving a process plan schema for a family of similar processes out of the process plan for an example plan in that family.

Using the rationale for determining what parts of a generalized process plan go into a schema, let us return to the question of what to schematize from the axle cover process plan. The upper portion of Figure 5-6 consists of features that enable the axle shaft to locate

on the center bore BZ* and run from one end of this bore to the other and put past face PZ+. The function of the axle cover is to protect the bearings (on which the axle shaft is mounted) and other components of the assembly from exposure to grit and other effects during operation of the vehicle. The part design of the axle cover must therefore ensure a snug fit that leaves no clearances (gaps) through which foreign particles may enter. A generalized description of the function of this part may be "a protective guard for the ends of a rotating shaft." Such a part must be mounted onto the shaft in some way and therefore requires threaded holes. Thus the features of PZ+, BZ*, PZ-1, H12 and PH fall into one functional group. Accordingly, the portion of the generalized plan that creates this set of features is packaged into a schema. Figure 5-11 shows the schema for these features, extracted from the generalized plan of the axle cover, based on the explanation process shown in Appendix 1. In the future, whenever LASIPP is required to create a process plan for components that incorporate this set of features in their design, it simply instantiates the axle cover schema thus obtaining the required process plan for this set of features. Thus, in one sweep LASIPP is able to specify the parameters for seven machining operations (the number of primitive operators in the axle cover schema), which in the absence of this schema, would have required searching through the knowledge base seven times. LASIPP finds the remaining operators for the complete process plan by searching the knowledge base for matching operators as in a generative planning system.

Insert Figure 5-11 about here

In summary, the above example reflects several desirable features of LASSIP as a process planning system. These features are: (1) LASSIP can "take advantage of similarity" by learning schemata that eliminate the wasteful redundant effort of repetitive process plan generation--a limitation of generative planning systems. More importantly, EBL provides the ability to make generalization of an example plan, so that the resulting process plan schema can be used for a family of process plans. (2) Unlike a variant planning system which only carries standard plans in its knowledge base, EBL gives LASSIP the flexibility of creating process plans operations-by-operation when necessary. Thus, if no schema (i.e., the standard plan in a variant system) is found applicable, LASSIP acts like a generative system, creating a process plan by concatenating a set of primitive operators. Generative and variant systems thus represent lower bounds of performance for LASIPP. (3) EBL allows chunking of knowledge, a distinguishing characteristic of human experts. While the initial schemata acquired by the system use primitive operators as their building blocks, over the course of time, the system will begin to acquire schemata that use other schemata as their building blocks (in Figure 5-11, schema 0002 uses schema 1175 as one of its components). This reflects the growth in the inferential powers of an intelligent system, demonstrated by its ability to "cover" increasingly greater "distances" in its problem state space with a single step (schema). (4) Since an EBL based system stores schemata and not

entire plans as in variant systems, there is less duplication of the stored knowledge. This is because the standardized plans are stored in the form of schemata, which are produced by generalizing existing plans so as to cover a family of similar processes. The variant systems developed to date have not been able to make such generalization systematically.

6 CONCLUDING REMARKS

The potential of using machine learning to enhance an intelligent process planning system is promising. Just as the process planning output of a human planner improves with the experience that the planner has had in creating plans for similar jobs in the past, LASIPP uses EBL to improve its performance progressively.

LASIPP is capable of generating plans based on part specifications. However, it can learn the patterns of plans and store them in the form of schemata. Therefore, when compared with generative systems, most important points on which LASIPP scores are the reduction in search space, intelligently guided search of this space, and the enhanced quality of plans produced due to the use of schemata.

In addition, the ability to generate schemata and save them for direct applications to similar process planning problems gives LASIPP the property of a variant system. The use of schemata also helps the standardization of process planning for similar parts. This feature enables LASIPP to integrate the generative and variant approaches of process planning. LASSIPP is the first process planning system that is equipped with EBL to integrate the features of both generative and

variant systems. In this paper, we have shown that there are plenty of advantages of having such an integrated approach based on EBL for process planning.

Acknowledgements

This research is supported in part by grants from the Office for Information Management, the Research Board, and Texas Instruments. We wish to thank Professor Gerald DeJong for providing us with his EBL program, which we used to implement a prototype of the learning unit for LASIPP. Thanks are also due to Texas Instruments and to IntelliCorp, who provide the Lisp machine and the software for developing LASIPP.

REFERENCES

Berenji, H. R., and Khoshnevis, B., "Use of Artificial Intelligence in Automated Process Planning," Computers in Mechanical Engineering, pp. 47-55, Sep. 1986.

Carbonell, J. G., "Learning by Analogy: Formulating and Generating Plans from Past Experience," in Michalski, R., et al. (Ed.), Machine Learning: An Artificial Intelligence Approach Vol. I, Tioga, pp. 83-134, 1983.

Chang, T. C., and Wysk, R. A., An Introduction to Automated Process Planning Systems, Prentice-Hall, Englewood Cliffs, NJ, 1985.

Chang, T. C., and Wysk, R. A., "Integrating CAD and CAM through Automated Process Planning," International Journal of Production Research, vol. 22, no. 5, pp. 877-894, 1984.

Davis, R., "Interactive Transfer of Expertise: Acquisition of New Inference Rules," Artificial Intelligence, Vol. 12, pp. 121-57, 1979.

DeJong, G. F., "An Approach to Learning from Observation," in Michalski, R., et al. (Ed.), Machine Learning: An Artificial Intelligence Approach Vol. II, Morgan Kaufmann, Los Altos, CA, pp. 571-590, 1986.

DeJong, G. F., Explanation Based Learning, Technical Report T-162, Artificial Intelligence Research Group, Coordinated Science Lab., Univ. of Illinois at Urbana-Champaign, July, 1985.

DeJong, G. F., and Mooney, R. J., "Explanation Based Learning: An Alternative View," Machine Learning, vol. 1, pp. 145-176, 1986.

Descotte, Y., and LaTombe, J. C., "GARI: A Problem Solver That Plans How Much to Machine Mechanical Parts," IJCAI 7, Vancouver, 1981.

Dietterich, T. G. and Michalski, R. S., "A Comparative Review of Selected Methods for Learning from Examples," in Machine Learning: An AI Approach, ed. R S. Michalski, J. G. Carbonell, and T. M. Mitchell, Tioga, 1983.

Eversheim, W., and Fuchs, H., "Integrated Generation of Drawings, Process Plans, and NC Tapes," in Advanced Manufacturing Technology, edited by P. Blake, North-Holland, 1980.

Fikes, R., Hart, P., Nilsson, N., "Learning and Executing Generalized Robot Plans," in Webber, B. L., and Nilsson, N. J., Reading in Artificial Intelligence, Tioga, Palo Alto, CA, pp. 231-249, 1981.

Holland, J., "Escaping Brittleness: The Possibility of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems," Machine Learning: An AI Approach, Michalski, Carbonell, and Mitchell (Eds.), Tioga Pub. Co., Palo Alto, CA, 1986.

Joshi, S., Chang, T. C., and Liu, R., "Process Planning Formalization in an AI Framework," International Journal for Artificial Intelligence in Engineering, Vol 1, No. 1, pp. 45-53, 1986.

Keller, R. M., "Defining Operationality for Explanation-Based Learning," Artificial Intelligence, Vol. 35, No. 2, pp. 227-242, 1988.

Langley, P., "Data-Driven Discovery of Physical Laws," Cognitive Science, Vol. 5, pp. 31-54, 1981.

Larkin, J., Mcdermott, J., Simon, D. P., Simon, H. A., "Expert and Novice Performance in Solving Physics Problems," Science, Vol. 208, June 20, 1980.

Link, C. H., "CAPP--CAM-I Automated Process Planning System," Proceedings of the 13th Annual Technical Meeting, Numerical Control Society, 1976.

Mann, W. S., Dunn, Jr., M. S., and Pfleder, S. J., Computerized Production Process Planning, Report #R77-942625-14, United Technologies Research Corp., 1977.

Mitchell, T. M., "Generalization as Search," in Webber, B. L., and Nilsson, N. J., Reading in Artificial Intelligence, Tioga, Palo Alto, CA, pp. 517-542, 1981.

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T., "Explanation Based Learning: A Unifying View," Machine Learning, vol. 1, pp. 47-80, 1986.

Nakazawa, H., and Suh, N. P., "Process Planning Based on Information Concept," Robotics & Computer-Integrated Manufacturing, vol. 1, no. 1, pp. 115-123, 1984.

Nau, D. S., and Chang, T. C., "Hierarchical Representation of Problem-Solving Knowledge in a Frame-Based Process Planning System," International Journal of Intelligent Systems, vol. 1, pp. 29-44, 1986.

Nau, D. S., and Chang, T. C., "Prospects for Process Selection Using Artificial Intelligence," Computers in Industry, vol. 4, pp. 253-263, 1983.

Nilsson, N. J., Principles of Artificial Intelligence, Tioga, Palo Alto, CA, 1980.

Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers II--Recent Progress," IBM J. of Research and Development, Vol. 11, No. 6, pp. 609-617, 1967.

Shaw, M., "Applying Inductive Learning to Enhance Knowledge-based Expert Systems," Decision Support Systems, Vol. 3, No. 4, pp. 319-332, 1987.

Shaw, M. J., "Knowledge-Based Scheduling in Flexible Manufacturing Systems: An Integration of Patern-Directed Inference and Heuristic Search," International Journal of Production Research (Special Issue on AI and Manufacturing), Vol. 26, No. 5, pp. 821-844, 1988.

Shaw, M., Menon, U., Park, S., "Applying Machine Learning to Knowledge-based Process Planning," in Expert Systems: Strategies & Solutions for Manufacturing Design and Planning, A. Kusiak (Ed.), Society of Manufacturing Engineers, Deerborn, MI, 1988.

TNO, "Introduction to MIPLAN," Organization for Industrial Research, Inc., Waltham, MA, 1981.

Winston, P. H., "Learning by Augmenting Rules and Accumulating Censors," in Michalski, R., et al. (Ed.), Machine Learning: An Artificial Intelligence Approach Vol. II, Morgan Kaufmann, Palo Alto, CA, pp. 45-62, 1986.

Wu, M. C., and Liu, C. R., "Automated Process Planning and Expert Systems," Proceedings of IEEE International Conference on Robotics and Automation, pp. 186-191, 1985.

Wysk, R. A., "An Automated Process Planning and Selection Program: APPAS," Ph.D. Thesis, School of Industrial Engineering, Purdue University, 1977.

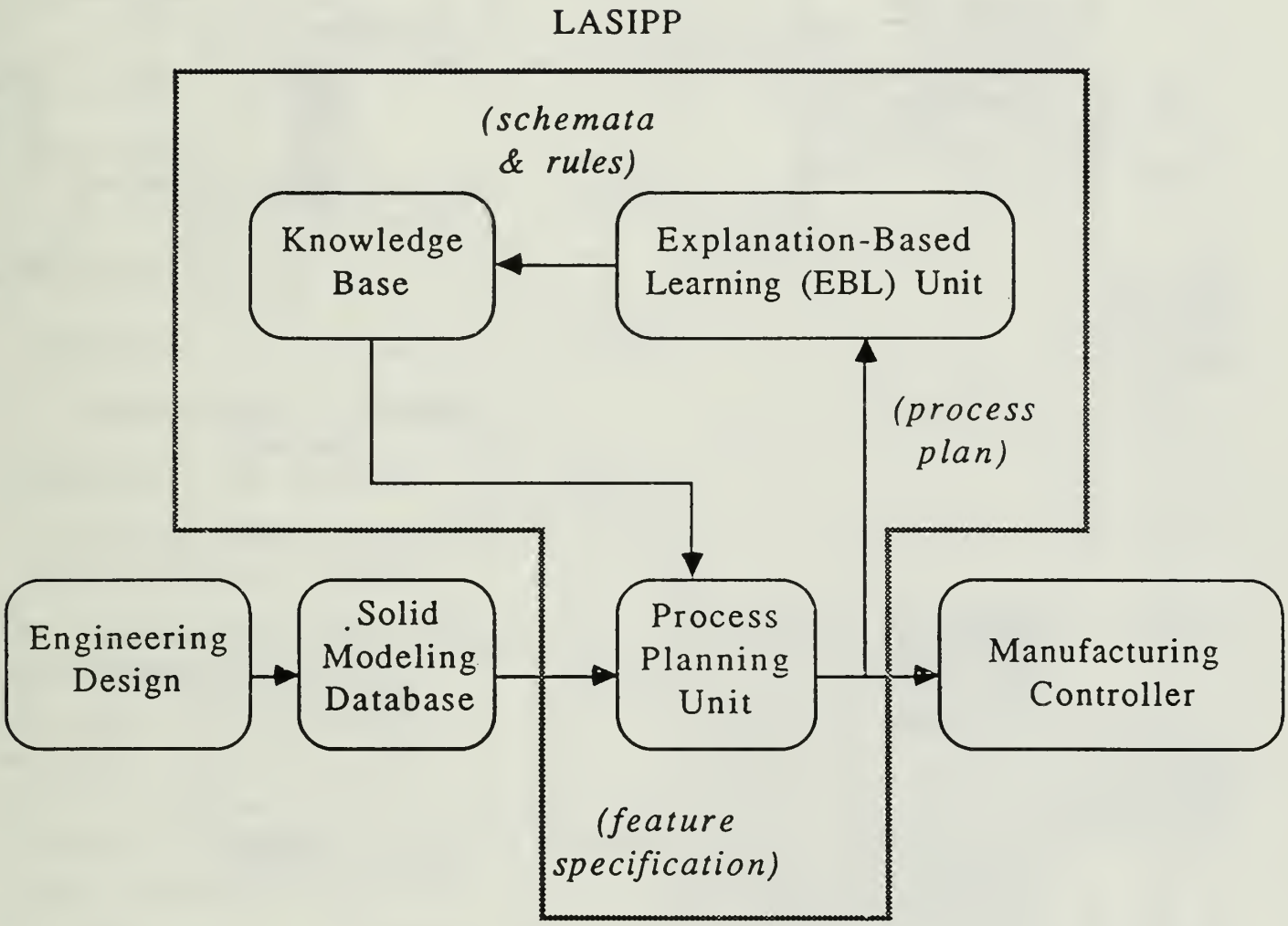


Figure 2-1 The Organization of LASIPP

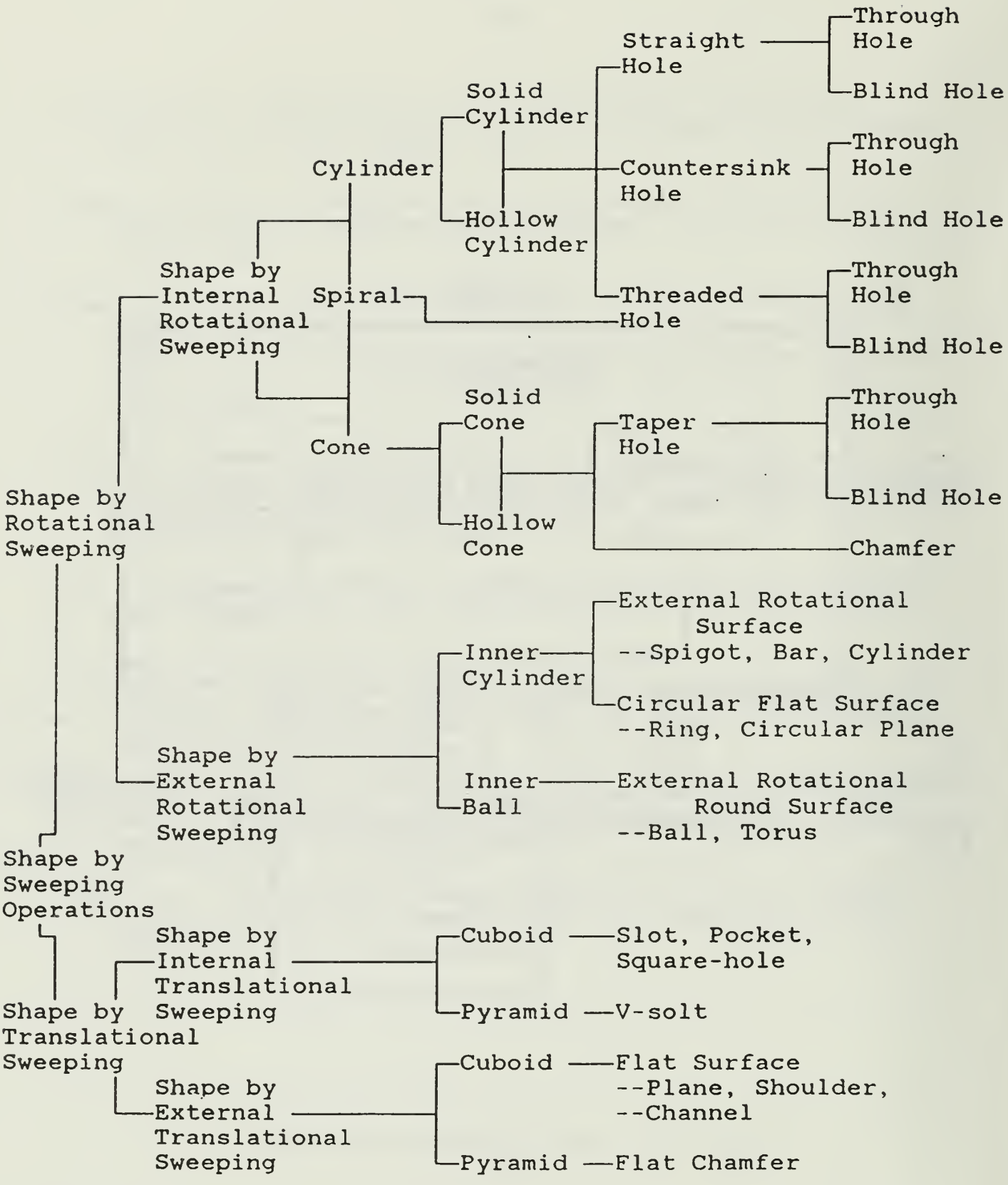
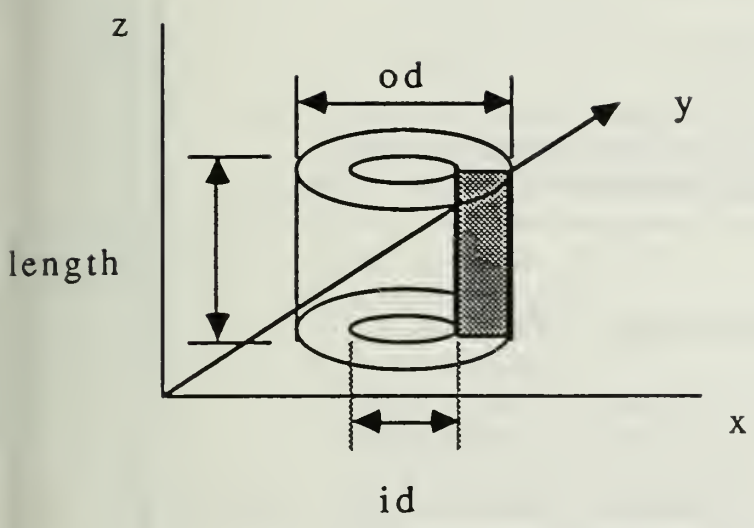
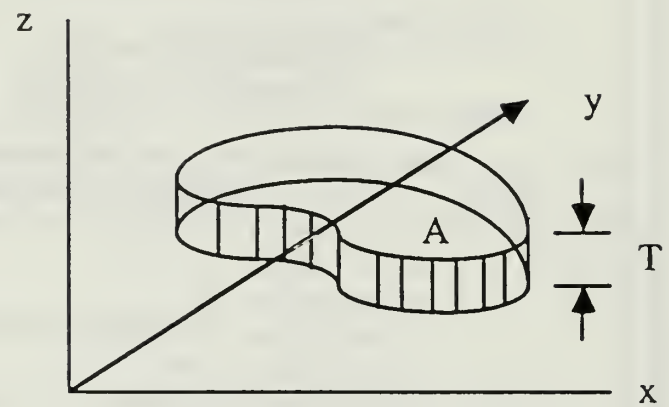


Figure 4-1 Hierarchy of Shapes to be Removed in Relation with Operations and Features



Cylinder (id, od, length)



Tsweep (area A * T)

The shaded rectangular area represents the cutting tool embedded in a cylindrical workpiece. By moving the tool in a circular motion, a volume of material, 'cylinder difference' is removed.

The area A represents a surface of the workpiece with amorphous shape. By moving the tool in a purely linear relative motion, a volume of material, 'tsweep difference', or translational difference is removed.

Figure 4-2 Shape Primitives

```
(rule /* rule name */ I1
      /* arguments */
         (?length1 ?width1 ?thick1 ?tol1 ?fs1)
      /* antecedent */
         (cuboid-difference
             ?length1 ?width1 ?thick1 ?tol1 ?fs1)
      /* consequent */
         (tsweep-difference
             ?length1 ?width1 ?thick1 ?tol1 ?fs1) )
```

I1 If: the shape to be removed is cuboid-difference,
Then: it is a specific type of tsweep-difference.

```
(rule /* rule name */ I2
      /* arguments */
         (?length2 ?width2 ?thick2 ?tol2 ?fs2)
      /* antecedent */
         (tsweep-removed
                 ?length2 ?width2 ?thick2 ?tol2 ?fs2)
      /* consequent */
         (flat-surface ?length2 ?width2 ?thick2 ?tol2 ?fs2))
```

I2 If: the removed shape is a tsweep-difference,
Then: a flat-surface is created.

```
(rule /* rule name */ I3
      /* arguments */
         (?id3 ?od3 ?length3 ?tol3 ?h3)
      /* antecedent */
         (cylinder-removed ?id3 ?od3 ?length3 ?tol3 ?h3)
      /* consequent */
         (straight-hole ?od3 ?length3 ?tol3 ?h3)  )
```

I3 If: the removed shape is a cylinder-difference
Then: a straight-hole is created

```
(rule /* rule name */ I4
      /* arguments */
         (?id4 ?od4 ?length4 ?pitch4 ?tol4 ?h4)
      /* antecedent */
         (mstraight-hole ?id4 ?length4 ?tol4 ?h4)
         (mspiral-removed
                  ?id4 ?od4 ?length4 ?pitch4 ?tol4 ?h4)
      /* consequent */
         (mthreaded-hole ?id4 ?od4 ?length4
                         ?pitch4 ?tol4 ?h4)  )
```

I4 If: the removed shapes are same size of several
cylinder-differences and spiral-differences,
Then: same size of multiple threaded holes are created.

Figure 4-3 Inference Rules in the Domain Theory

```
(rule /* rule name */ I5
      /* arguments */
         (?id5 ?max-depth5 ?length5 ?tol5 ?cfs5)
      /* antecedent */
         (inner-cylinder-removed ?id5 ?max-depth5 ?length5
                ?tol5 ?cfs5)    (less-than ?length5 0.1)
      /* consequent */
         (circular-flat-surface ?id5 ?max-depth5 ?length5
                                ?tol5 ?cfs5)  )
```

I5 If: the removed shape is a inner-cylinder-difference with length less than 0.1 inch
Then: a circular-flat-surface is created

```
(rule /* rule name */ I6
      /* arguments */
         (?id6 ?od6 ?length6 ?tol6 ?h6)
      /* antecedent */
         (mcylinder-removed ?id6 ?od6 ?length6 ?tol6 ?h6)
      /* consequent */
         (mstraight-hole ?od6 ?length6 ?tol6 ?h6)  )
```

I6 If: the removed shapes are same size of several cylinder-differences,
Then: same size of multiple straight holes are created.

```
(rule /* rule name */ I7
      /* arguments */
       (?id7 ?od7 ?cid7 ?cod7 ?length7 ?clength7 ?tol7 ?h7)
      /* antecedent */
         (cylinder-removed ?id7 ?od7 ?length7 ?tol7 ?h7)
         (cylinder-removed ?od7 ?cod7 ?clength7 ?tol7 ?h7)
      /* consequent */
         (countersink-hole ?od7 ?cod7 ?length7 ?clength7
                           ?tol7 ?h7)  )
```

I7 If: the removed shapes are two different cylinder-differences connected with each other,
Then: a countersink-hole is created.

```
(rule /* rule name */ I8
      /* arguments */
         (?id8 ?max-depth8 ?width8 ?length8 ?tol8 ?cfs8)
      /* antecedent */
         (circular-flat-surface ?id8 ?max-depth8
          ?length8 ?tol8 ?cfs8) (sum ?width81 ?id8 ?width8)
         (product 2 ?max-depth8 ?width81)
      /* consequent */
        (flat-surface ?width8 ?width8 ?length8 ?tol8 ?cfs8))
```

I8 If: the feature is a circular flat surface,
Then: it is a specific type of a flat-surface.

Figure 4-3 Inference Rules in the Domain Theory (Continued)

```
operator Twist Drilling (TD)
/* argument list */
     (?id-td ?od-td ?length-td ?tol-td ?h-td ...)
/* antecedent */
     flat-surface (?length-td ?width-td ?thick-td ?tol-td
                   ?fs-td)
     cylinder-difference
          (?id-td ?od-td ?length-td ?tol-td ?h-td)
     exist-drill-size (?dtl-td)  placed (?h-td ?fs-td)
     /* operation boundary conditions */
     greater-than (?od-td 0.063)
     greater-than (2 ?od-td) ...
/* consequent */
     cylinder-removed
          (?id-td ?od-td ?length-td ?tol-td ?h-td)
```

```
operator Rough-Finish Milling (RFM)
/* argument list */
     (?length-rfm ?width-rfm ?thick-rfm ?fs-rfm ...)
/* antecedent */
     tsweep-difference
     (?length-rfm ?width-rfm ?thick-rfm ?tol-rfm ?fs-rfm)
     exist-cutter-size (?dtl-rfm)
     /* operation boundary condition */
     greater-than (?length-rfm 0)
     greater-than (50 ?length-rfm) ...
/* consequent */
     tsweep-removed (?length-rfm ?width-rfm ?thick-rfm
                     ?tol-rfm ?fs-rfm)
```

```
operator Rough-Finish Bore (RFB)
/* argument list */
     (?id-rfb ?od-rfb ?length-rfb ?tol-rfb ?h-rfb ...)
/* antecedent */
     flat-surface (?length-rfb ?width-rfb ?thick-rfb
                   ?tol-rfb ?fs-rfb)
     cylinder-difference
          (?id-rfb ?od-rfb ?length-rfb ?tol-rfb ?h-rfb)
     placed (?h-rfb ?fs-rfb)
     /* operation boundary conditions */
     dif (?od-rfb ?id-rfb ?depth-rfb)
     less-than-or-equal (?depth-rfb 1)
     greater-than (?id-rfb 3) ...
/* consequent */
     cylinder-removed
          (?id-rfb ?od-rfb ?length-rfb ?tol-rfb ?h-rfb)
```

Figure 4-4 Operators in the Domain Theory

```
operator Rough Bore External (RBE)
/* argument list */
     (?id-rbe ?length-rbe ?tol-rbe ?cfs-rbe ...)
/* antecedent */
     inner-cylinder-difference (?id-rbe ?max-depth-rbe
                          ?length-rbe ?tol-rbe ?cfs-rbe)
     /* operation boundary conditions */
     ... ... ...
/* consequent */
     inner-cylinder-removed (?id-rbe ?max-depth-rbe
                          ?length-rbe ?tol-rbe ?cfs-rbe)
```

```
operator Multi Pattern Drilling (DMP)
/* argument list */
     (?id-dmp ?od-dmp ?length-dmp ?tol-dmp ?h-dmp ...)
/* antecedent */
     flat-surface (?length-dmp ?width-dmp ?thick-dmp
                   ?tol-dmp ?fs-dmp)
     mcylinder-difference
          (?id-dmp ?od-dmp ?length-dmp ?tol-dmp ?h-dmp)
     exist-drill-size (?dtl-dmp)  placed (?h-dmp ?fs-dmp)
     /* operation boundary conditions */
     ... ... ...
/* consequent */
     mcylinder-removed
          (?id-dmp ?od-dmp ?length-dmp ?tol-dmp ?h-dmp)
```

```
operator Multi Pattern Tapping (TMP)
/* argument list */
     (?id-tmp ?od-tmp ?length-tmp ?tol-tmp ?h-tmp ...)
/* antecedent */
     mspiral-difference (?id-tmp ?od-tmp ?length-tmp
                         ?pitch-tmp ?tol-tmp ?h-tmp)
     exist-tap-size (?dtl-tmp)
     /* operation boundary conditions */
     ... ... ...
/* consequent */
   mspiral-removed
   (?id-tmp ?od-tmp ?length-tmp ?pitch-tmp ?tol-tmp ?h-tmp)
```

Figure 4-4 Operators in the Domain Theory (Continued)

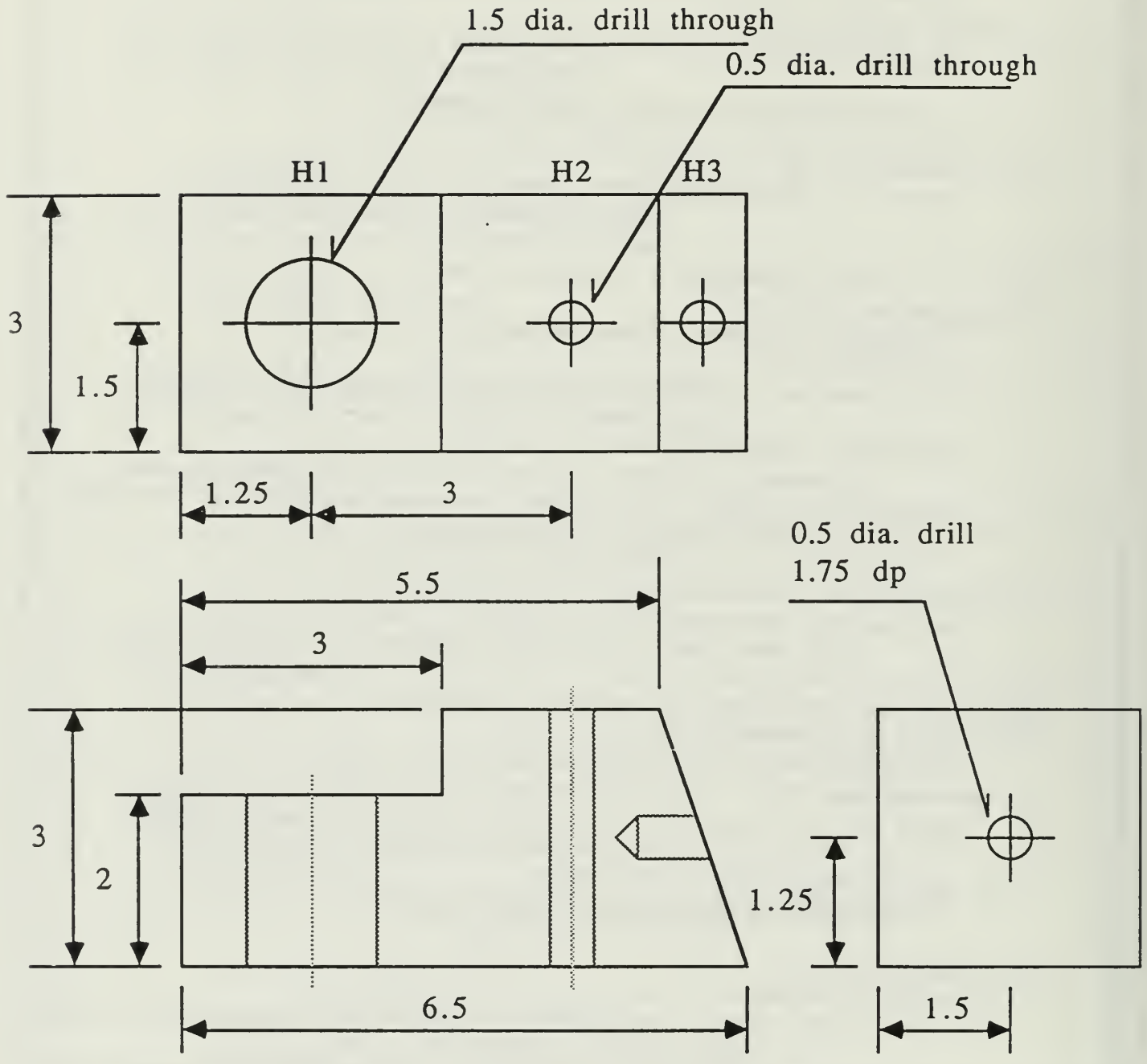


Figure 4-5 Engineering Design of the Example Part

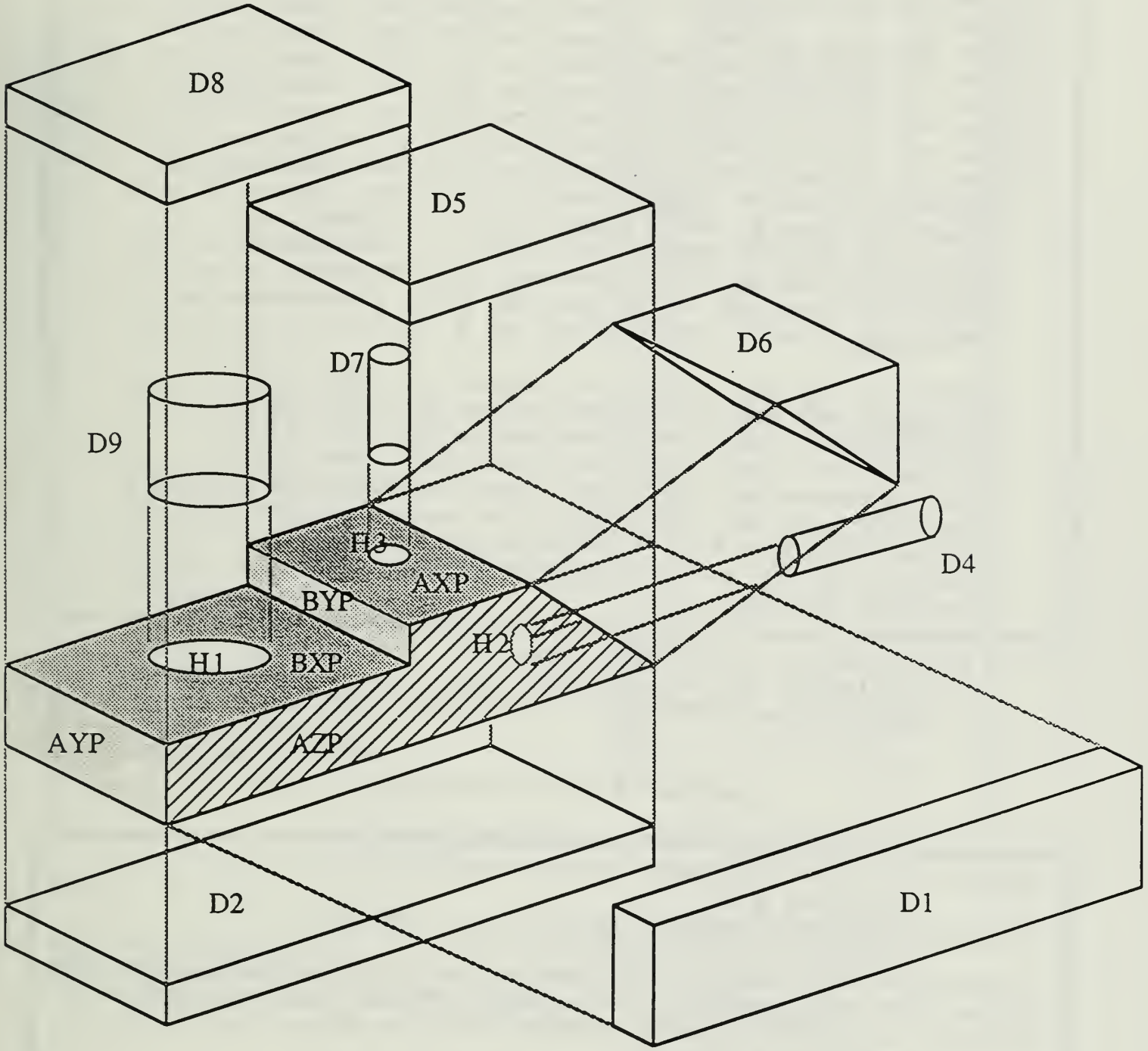


Figure 4-6 A Pictorial Depiction of Material Removal -- the Finished Part is Created from a Block by Removal of Volumes D1 - D9

Goal State Description

```
(ftype AXP FLAT-SURFACE) (direction AXP XP) (quality AXP .015)
(ftype BXP FLAT-SURFACE) (direction BXP XP) (quality BXP .015)
(ftype AXN FLAT-SURFACE) (direction AXN XN) (quality AXN .015)
(ftype AYP FLAT-SURFACE) (direction AYP YP) (quality AYP .05)
(ftype BYP FLAT-SURFACE) (direction BYP YP) (quality BYP .015)
(ftype BYQ FLAT-SURFACE) (direction BYQ YQ) (quality BYQ .015)
(ftype AZP FLAT-SURFACE) (direction AZP ZP) (quality AZP .015)
(ftype AZN FLAT-SURFACE) (direction AZN ZN) (quality AZN .015)
   (ftype H1 STRAGHT-HOLE) (dia H1 1.5) (tolerance H1 .01)
   (starting-from H1 BXP) (opening-into H1 AXN)
   (ftype H3 STRAGHT-HOLE) (dia H3 0.5) (tolerance H3 .01)
   (starting-from H3 AXP) (opening-into H3 AXN)
   (ftype H2 STRAIGHT-HOLE) (dia H2 0.5) (tolerance H2 .01)
   (starting-from H2 BYQ) (depth H2 1.75)

   (distance AXP AXN 3.00) (tolerance AXP AXN 0.05)
   (distance BXP AXN 2.00) (tolerance BXP AXN 0.05)
   (distance AZP AZN 3.00) (tolerance AZP AZN 0.05)
   (distance AYP BYP 3.00) (tolerance AYP BYP 0.05)
   *(min-distance AYP BYQ 5.50) (tolerance AYP BYQ 0.05)
   *(max-distance AYP BYQ 6.50) (tolerance AYP BYQ 0.05)
   (distance H1 AYP 1.25) (distance H1 AZP 1.50)
   (distance H1 H3 3.00) (distance H3 AZP 1.50)
   (distance H2 AXN 1.25)
```

* Since BYQ is an angular face, two dimensions ´min-distance´ & ´max-distance´ are required to specify its relative position w.r.t. vertical face AYP

Initial State Description

```
(ftype B´XP FLAT-SURFACE) (direction B´XP XP)
        (quality B´XP .05)
(ftype A´XN FLAT-SURFACE) (direction A´XN XN)
        (quality A´XN .05)
(ftype AYP FLAT-SURFACE) (direction AYP YP) (quality AYP .05)
(ftype AYN FLAT-SURFACE) (direction AYN YN) (quality AYN .05)
(ftype A´ZP FLAT-SURFACE) (direction A´ZP ZP)
        (quality A´ZP .05)
(ftype A´ZN FLAT-SURFACE) (direction A´ZN ZN)
        (quality A´ZN .05)

   (distance B´XP A´XN 3.20) (tolerance B´XP A´XN 0.05)
   (distance A´ZP A´ZN 3.20) (tolerance AZP AZN 0.05)
   (distance AYP AYN 6.7) (tolerance AYP AYN 0.05)
```

Figure 4-7 Feature Specification of Initial and Goal States Using Form Feature Language

PROCESS PLAN

Activity (Operator)	Parameter Value	Difference Removed
FACE-TO-REST-ON	A´ZN	
PLACE-ON-MACHINE	MILL	
FINISH-MILL	A´ZP	(D1)
FACE-TO-REST-ON	A´XP	
PLACE-ON-MACHINE	MILL	
FINISH-MILL	A´XN	(D2)
FACE-TO-REST-ON	AZP	
PLACE-ON-MACHINE	MILL	
FINISH-MILL	A´ZN	(D3)
FACE-TO-REST-ON	AYP	
PLACE-ON-MACHINE	DRILL	
TWIST-DRILL	H2	(D4)
FACE-TO-REST-ON	AXN	
PLACE-ON-MACHINE	MILL	
FINISH-MILL	A´XP	(D5)
FACE-TO-REST-ON	AZP	
PLACE-ON-MACHINE	MILL	
ROUGH-FINISH-MILL	A´YN	(D6)
FACE-TO-REST-ON	AXN	
PLACE-ON-MACHINE	DRILL	
TWIST-DRILL	H3	(D7)
PLACE-ON-MACHINE	MILL	
ROUGH-FINISH-MILL	BXP	(D8)
PLACE-ON-MACHINE	DRILL	
TWIST-DRILL	H1	(D9)

Note:

AZP (A: convex cube, Z: axis direction, P: positive side)
BXN (B: concave cube, X: axis direction,
N: negative (hidden) side)
A´ZP: initial surface of AZP before the process

Figure 4-8 Process Plan for Making the Example Part

GIVEN

o DOMAIN THEORY

1) Inference Rules
(I1, I2, I3)
2) Operators
TD: TWIST DRILLING
RFM: ROUGH-FINISH MILLING

o GOAL

A block with a Shoulder on which a Hole locates

o INITIAL STATE

A block

o OBSERVED OPERATOR SEQUENCE

RFM (length = 3, width = 3, thickness = 1,
tool diameter = 1.0, tolerence = 0.015)
TD (inner-diameter = 0, outter-diameter = 1.5,
length = 2, tool diameter = 1.5,
tolerence = 0.01)

DETERMINE

A general schema for the Block-with-Hole

Figure 5-1 Specification of the Block-with-Hole Learning Problem

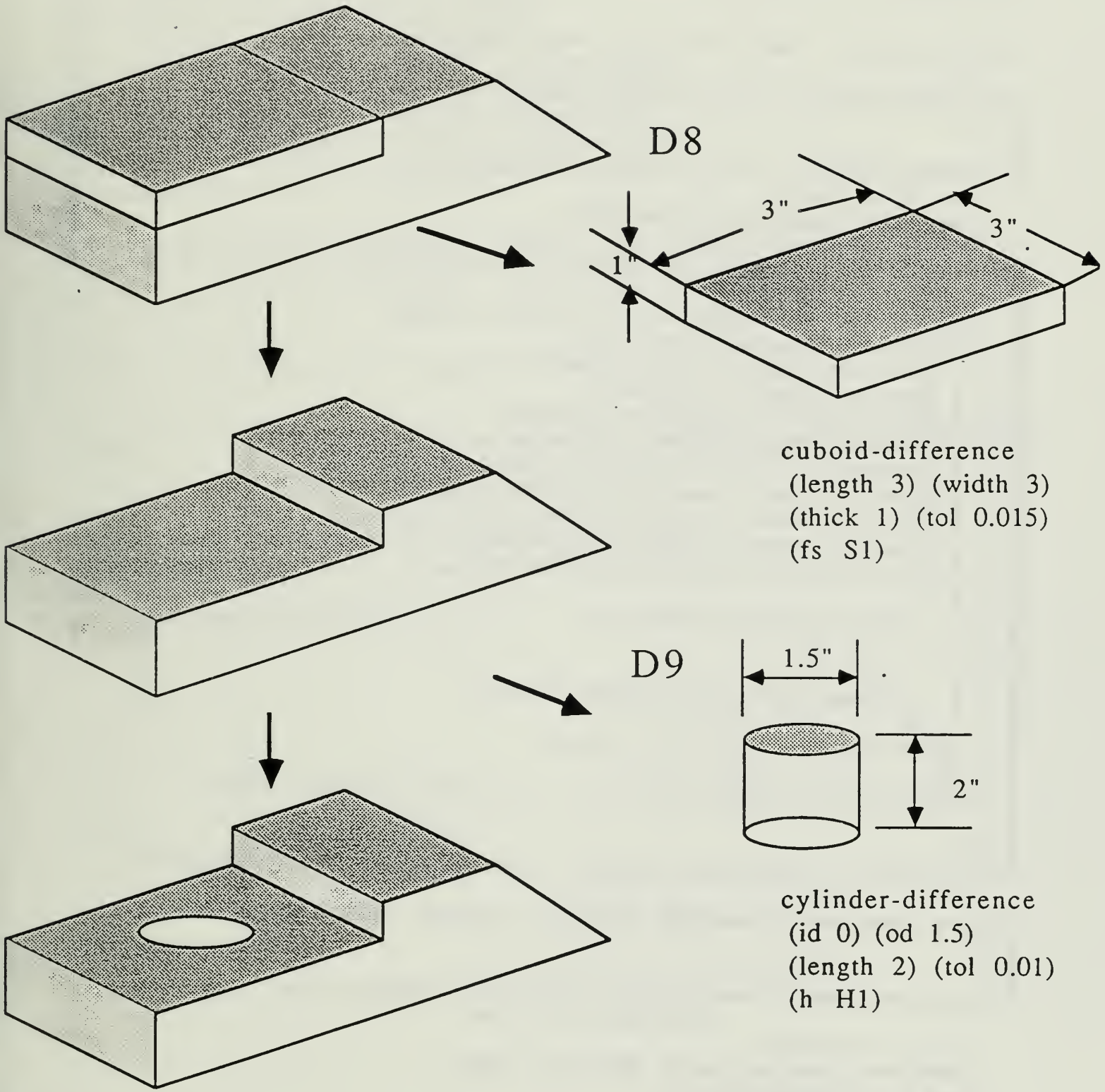


Figure 5-2 Internal Representation of D8 and D9

```
straight-hole (1.5 2 0.01 H1) :goal
    | U1
    |
straight-hole (?od3 ?length3 ?tol3 ?h3) :rule I3
   <--
cylinder-removed (?id3 ?od3 ?length3 ?tol3 ?h3)
|| U2
||
cylinder-removed (?id-td ?od-td ?length-td ?tol-td ?h-td)
   <== e (effect)
   operator Twist Drilling (TD)
   <== p (precondition)
flat-surface (?length-td ?width-td ?thick-td ?tol-td ?fs-td)
|| cylinder-difference
|| (?id-td ?od-td ?length-td ?tol-td ?h-td)
||  | exist-drill-size (?dtl-td)
||  |  | greater-than (?od-td 0.063)
||  |  | greater-than (2 ?od-td)
||  |  | dif (?od-td ?id-td ?thick-td)
||  |  | greater-than (?thick-td 0.1)
||  |  | product (12 ?od-td ?max-length-td)
||  |  | greater-than (?max-length-td ?length-td)
||  |  | greater-than-or-equal (?tol-td 0.01)
||  |  | numeric-equal (?dtl-td ?od-td)
||  |  | dif (?od-td ?dtl-td ?remain-td)
||  |  |
||  |  |   placed (?h-td ?fs-td)
||  |  |       | U3
||  |  |       |
||  |  |   placed (H1 S1) :initial state
||  |  | U4
||  | exist-drill-size (1.5) :fact
||  | U5
|| cylinder-difference (0 1.5 2 0.01 H1) :initial state
||
|| U6
flat-surface (?length2 ?width2 ?thick2 ?tol2 ?fs2) :rule I2
   <--
tsweep-removed (?length2 ?width2 ?thick2 ?tol2 ?fs2)
|| U7
||
tsweep-removed (?length-rfm ?width-rfm ?thick-rfm
                    ?tol-rfm ?fs-rfm)
   <== e (effect)
   operator Rough-Finish Milling (RFM)
   <== p (precondition)
tsweep-difference
||    (?length-rfm ?width-rfm ?thick-rfm ?tol-rfm ?fs-rfm)
|| exist-cutter-size (?dtl-rfm)
|| | greater-than (?length-rfm 0)
```

Figure 5-3 Explanation Structure for the Block-with-Hole Problem

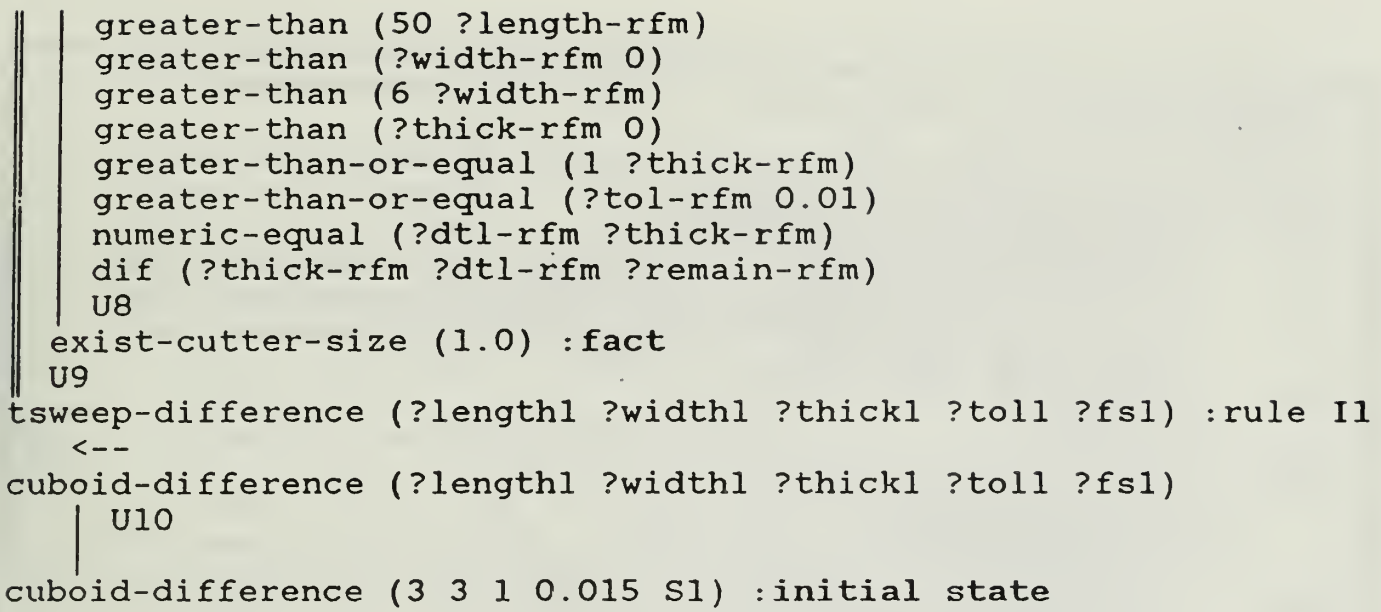


Figure 5-3 Explanation Structure for the Block-with-Hole Problem (Continued)

note:
——— Specific Unification only
═══ Both Specific and General Unifications

U# : Unification Number

Unific -ation	Specific (piecewise) S	General (piecewise) G
U1	Goal (H1/?h3, 1.5/?od3, 2/?length3, 0.01/?tol3)	
U2	Subgoal (H1/?h-td, 1.5/?od-td, 0.01/?tol-td 2/?length-td)	?h-td/?h3, ?tol-td/?tol3, ?length-td/?length3, ?id-td/?id3, ?od-td/?od3
U3	H1/?h-td, S1/?fs-td, H1/?h3	
U4	1.5/?dtl-td, Subgoal (1.5/?od-td), 0/?remain-td, 0/?remain3	
U5	0/?id-td, 1.5/?od-td, 2/?lenghth-td, 0.01/?tol-td, H1/?h-td, 0.01/?tol3	
U6	Subgoal (S1/?fs2)	?fs2/?fs-td, ?tol2/?tol-td, ?length2/?length-td, ?width2/?width-td, ?thick2/?thick-td
U7	Subgoal (S1/?fs-rfm)	?fs-rfm/?fs2, ?tol-rfm/?tol2, ?length-rfm/?length2, ?width-rfm/?width2, ?thick-rfm/?thick2
U8	1.0/?dtl-rfm, Subgoal (1.0/?thick-rfm), 0/?remain-rfm, 0/?remain2	
U9	Subgoal (S1/?fs1)	?length1/?length-rfm, ?width1/?width-rfm, ?thick1/?thick-rfm, ?tol1/?tol-rfm, ?fs1/?fs-rfm
U10	3/?length1, 3/?width1, 1/?thick1, 0.015/?tol1, S1/?fs1, 3/?length-rfm, 3/?width-rfm, 1/?thick-rfm, 0.015/?tol-rfm, S1/?fs-rfm	

Figure 5-4 The Substitution Lists G and S Obtained by Unification

Final G (general Substitution) List:

```
G = {?h-td/?h3, ?tol-td/?tol3, ?id-td/?id3, ?od-td/?od3,
     ?length-td/?length3, ?fs2/?fs-td, ?tol2/?tol-td,
     ?length2/?length-td, ?width2/?width-td, ?thick2/?thick-td,
     ?fs-rfm/?fs2, ?tol-rfm/?tol2, ?length-rfm/?length2,
     ?width-rfm/?width2, ?thick-rfm/?thick2,
     ?length1/?length-rfm, ?width1/?width-rfm,
     ?thick1/?thick-rfm, ?tol1/?tol-rfm, ?fs1/?fs-rfm}
```

Final S (specific substitution) List:

```
S = {H1/?h-td, S1/?fs-td, H1/?h3, 1.5/?dtl-td, 0/?remain-td,
     0/?remain3, 0/?id-td, 1.5/?od-td, 2/?lenghth-td,
     0.01/?tol-td, H1/?h-td, 0.01/?tol3, 1.0/?dtl-rfm,
     0/?remain-rfm, 0/?remain2, 3/?length1, 3/?width1,
     1/?thick1, 0.015/?tol1, S1/?fs1, 3/?length-rfm,
     3/?width-rfm, 1/?thick-rfm, 0.015/?tol-rfm, S1/?fs-rfm}
```

Figure 5-4 The Substitution Lists G and S Obtained by Unification (Continued)

```
(SCHEMA

/* schema name */
     SCHEMA1175

/* argument list */
     (?length-rfm ?width-rfm ?fs-rfm ... ?id-td ?od-td)

/* antecedent */
     (tsweep-difference ?length-rfm ?width-rfm
          ?thick-rfm ?tol-rfm ?fs-rfm)
     (cylinder-difference ?id-td ?od-td ?length-td
          ?tol-td ?h-td)
     (placed ?h-td ?fs-rfm)
     (exist-drill-size ?dtl-td)
     (exist-cutter-size ?dtl-rfm)

     /* operation boundary conditions */
     (greater-than ?lengh-rfm 0)
     ......
     (dif ?od-td ?dtl-td ?remain-td)
     (numeric-equal ?remain-td 0)

/* consequent */
     (straight-hole ?od-td ?length-td ?tol-td ?h-td)
     (flat-surface ?length-rfm ?width-rfm
                   ?thick-rfm ?tol-rfm ?fs-rfm)

/* generalized process plan */
     Rough-Finish Milling
     Twist Drilling
```

Figure 5-5 The Schema for the Learning Example

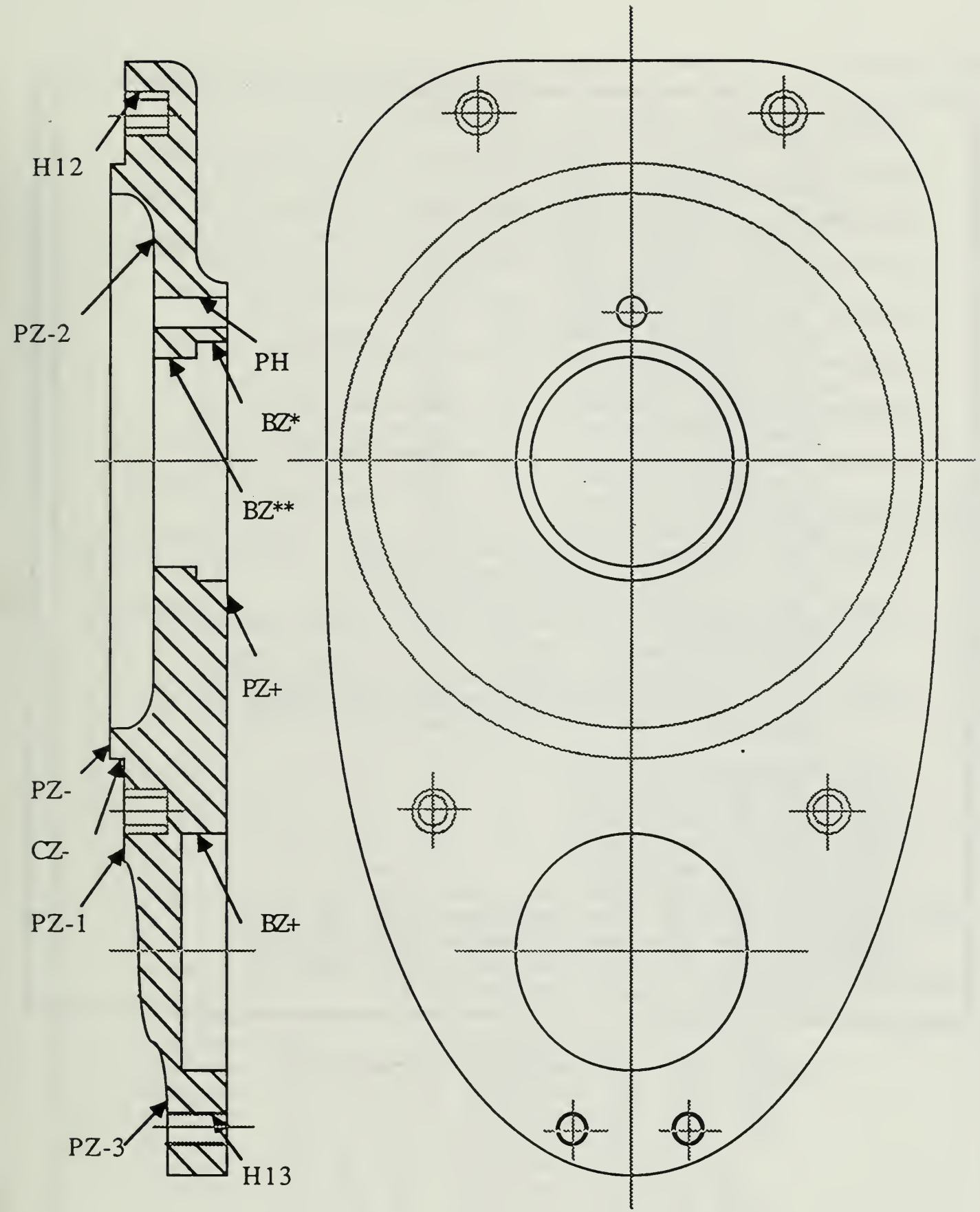


Figure 5-6 Engineering Design for the Axle Cover

```
(ftype PZ+ FLAT-SURFACE) (direction PZ+ ZP)
     (quality PZ+ 0.08)
(ftype PZ- CIRCULAR-FLAT-SURFACE) (direction PZ- ZN)
     (quality PZ- 0.08)
(ftype PZ-1 CIRCULAR-FLAT-SURFACE) (direction PZ-1 ZN)
     (quality PZ-1 0.08)
(ftype PZ-2 CIRCULAR-FLAT-SURFACE) (direction PZ-2 ZN)
     (quality PZ-2 0.08)
(ftype PZ-3 FLAT-SURFACE) (direction PZ-3 ZN)
     (quality PZ-3 0.08)
(ftype PYP FLAT-SURFACE) (direction PYP YP)
     (quality PYP 0.08)
(ftype PYN FLAT-SURFACE) (direction PYN YN)
     (quality PYN 0.08)
(ftype PXP FLAT-SURFACE) (direction PXP XP)
     (quality PXP 0.08)
(ftype PXN FLAT-SURFACE) (direction PXN XN)
     (quality PXN 0.08)
(ftype CZ- EXTERNAL-ROTATIONAL-SURFACE) (direction CZ- ZN)
     (dia CZ- 232) (quality CZ- 0.08)
(ftype BZ* COUNTERSINK-HOLE) (dia BZ* 71)
     (countersink-dia BZ* 79) (countersink-depth BZ* 10)
     (starting from BZ* PZ+)
     (opening-into BZ* PZ-2) (quality BZ* 0.08)
(ftype BZ+ STRAIGHT-HOLE) (dia BZ+ 79) (depth H2 15)
     (starting-from BZ+ PZ+) (quality H2 .08)

(distance PZ+ PZ-1 35)
(distance PZ-1 PZ- 5)
(distance PZ+ PZ-2 32)
(distance PZ+ PZ-3 20)
(distance PYP PYN 350)
(distance PXP PXN 250)
(distance BZ* PYP 150) (distance BZ* PXP 125)
(distance BZ* BZ+ 165) (distance BZ+ PXP 125)
(distance CZ- PYP 230) (distance CZ- PXP 125)
* all distance-tolerances are 0.01
```

Figure 5-7 Initial State Description

```
(ftype PZ+ FLAT-SURFACE) (direction PZ+ ZP)
     (quality PZ+ 0.008)
(ftype PZ- CIRCULAR-FLAT-SURFACE) (direction PZ- ZN)
     (quality PZ- 0.001)
(ftype PZ-1 CIRCULAR-FLAT-SURFACE) (direction PZ-1 ZN)
     (quality PZ-1 0.001)
(ftype PZ-2 CIRCULAR-FLAT-SURFACE) (direction PZ-2 ZN)
     (quality PZ-2 0.08)
(ftype PZ-3 FLAT-SURFACE) (direction PZ-3 ZN)
     (quality PZ-3 0.08)
(ftype PYP FLAT-SURFACE) (direction PYP YP)
     (quality PYP 0.08)
(ftype PYN FLAT-SURFACE) (direction PYN YN)
     (quality PYN 0.08)
(ftype PXP FLAT-SURFACE) (direction PXP XP)
     (quality PXP 0.08)
(ftype PXN FLAT-SURFACE) (direction PXN XN)
     (quality PXN 0.08)
(ftype CZ- EXTERNAL-ROTATIONAL-SURFACE) (direction CZ- ZN)
     (chamfer-width CZ- 1.2) (chamfer-depth CZ- 1.2)
     (dia CZ- 230) (quality CZ- 0.001)
(ftype BZ* COUNTERSINK-HOLE) (dia BZ* 72)
     (countersink-dia BZ* 80) (countersink-depth BZ* 10)
     (chamfer-width BZ* 1.2) (chamfer-depth BZ* 1.2)
     (countersink-chamfer-width BZ* 1.2)
     (countersink-chamfer-depth BZ* 1.2)
     (starting from BZ* PZ+)
     (opening-into BZ* PZ-2) (quality BZ* 0.001)
(ftype BZ+ STRAIGHT-HOLE) (dia BZ+ 80) (depth H2 15)
     (chamfer-width BZ+ 1.2) (chamfer-depth BZ+ 1.2)
     (starting-from BZ+ PZ+) (quality H2 .01)
(ftype PH STRAIGHT-HOLE) (dia PH 10) (quality H1 .01)
     (starting-from PH PZ+) (opening-into PH PZ-2)
(ftype H13 MTHREADED-HOLE) (dia H13 10)
     (thread-dia H13 12) (pitch H13 1) (quality H13 0.001)
     (starting-from H13 PZ+) (opening-into H13 PZ-3)
(ftype H12 MTHREADED-HOLE) (dia H12 12) (depth H12 18)
     (thread-dia H12 14) (pitch H12 1) (quality H12 0.001)
     (starting-from H12 PZ-1)
(distance PZ+ PZ-1 35) (distance PZ-1 PZ- 5)
(distance PZ+ PZ-2 32) (distance PZ+ PZ-3 20)
(distance PYP PYN 350) (distance PXP PXN 250)
(distance BZ* PYP 150) (distance BZ* PXP 125)
(distance BZ* BZ+ 165) (distance BZ+ PXP 125)
(distance BZ* PH 50) (distance PH PXP 125)
(distance BZ+ H13 60) (distance H13 PXP 20)
(distance H13 PXN 20) (distance CZ- PYP 150)
(distance CZ- PXP 125) (distance CZ- H12 125)
(distance H12 PXP 20) (distance H12 PXN 20)
** all distance-tolerances are 0.001
```

Figure 5-8 Goal State Description of the Axle Cover Specifications

```
D1    (tsweep-difference 350 250 0.125 0.008 PZ+)
D2    (cylinder-difference 71 72 22 0.001 BZ*)
D3    (cylinder-difference 79 80 10 0.001 BZ*)
D6    (cylinder-difference 0 10 32 0.01 PH)
D12   (inner-cylinder-difference 230 30 0.01 0.001 PZ-1)
D14   (mcylinder-difference 0 12 18 0.001 H12)
D15   (mspiral-difference 12 14 18 1 0.001 H12)
```

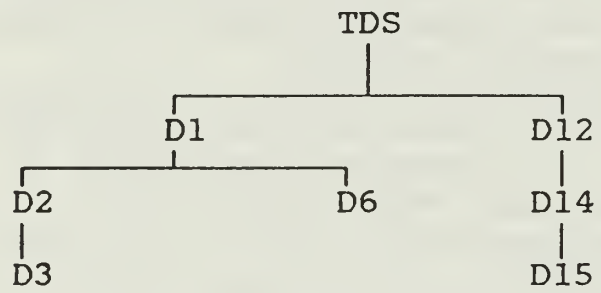


Figure 5-9 Shapes to be removed and Their Partial Ordering

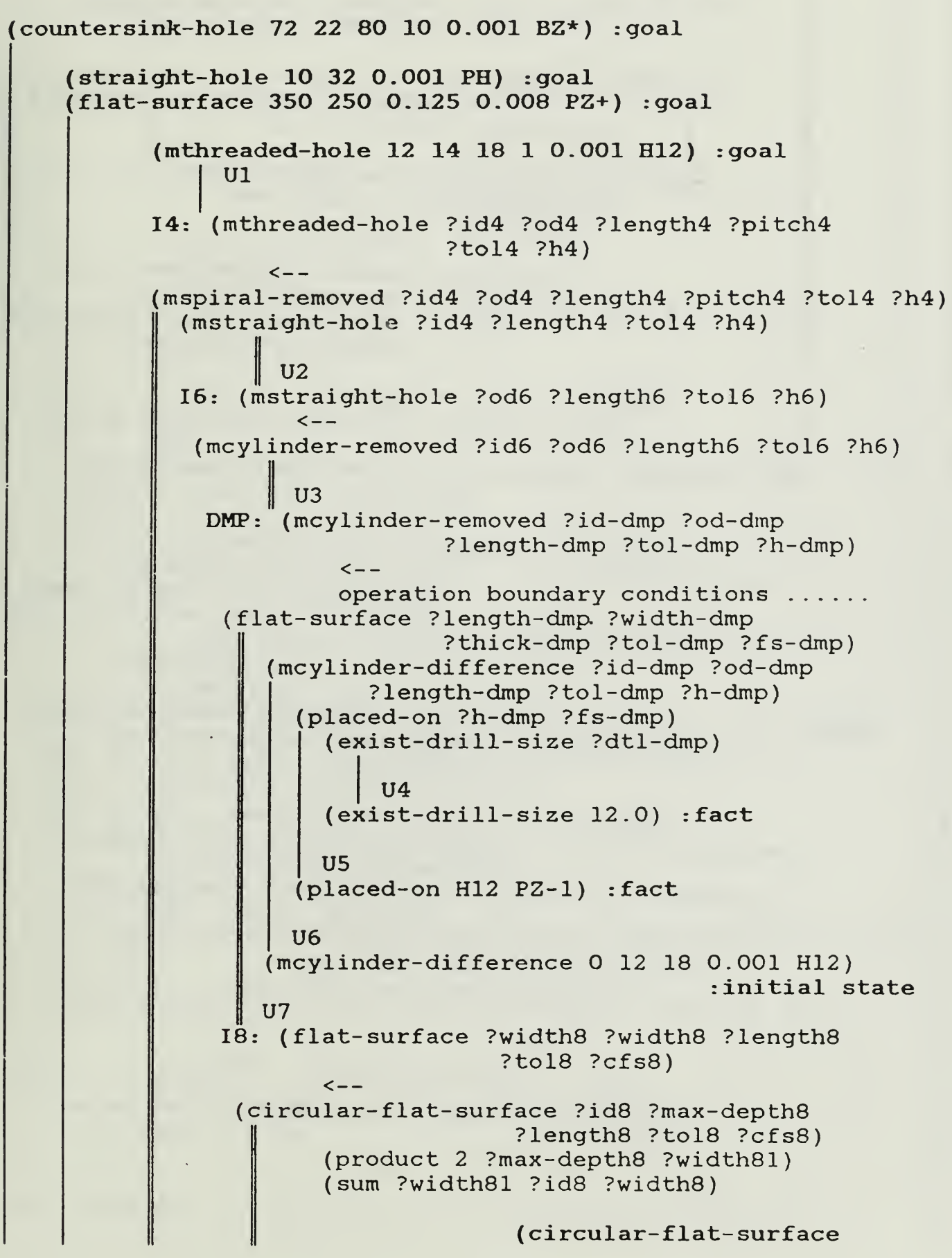
Activity (Operator)	Parameter value	Difference Removed
Face-to-rest-on	PZ-	
Place-on-machine	Mill	
Rough-finish-mill	PZ+	D1
Place-on-machine	Bore	
Rough-finish-bore	BZ*	D2
Rough-finish-bore	BZ*	D3
Place-on-machine	Drill	
Twist-drill	PH	D6
Face-to-rest-on	PZ+	
Place-on-machine	Bore	
Rough-bore-external	PZ-1	D12
Place-on-machine	Drill	
Drill-multi-pattern	H12	D14
Tap-multi-pattern	H12	D15

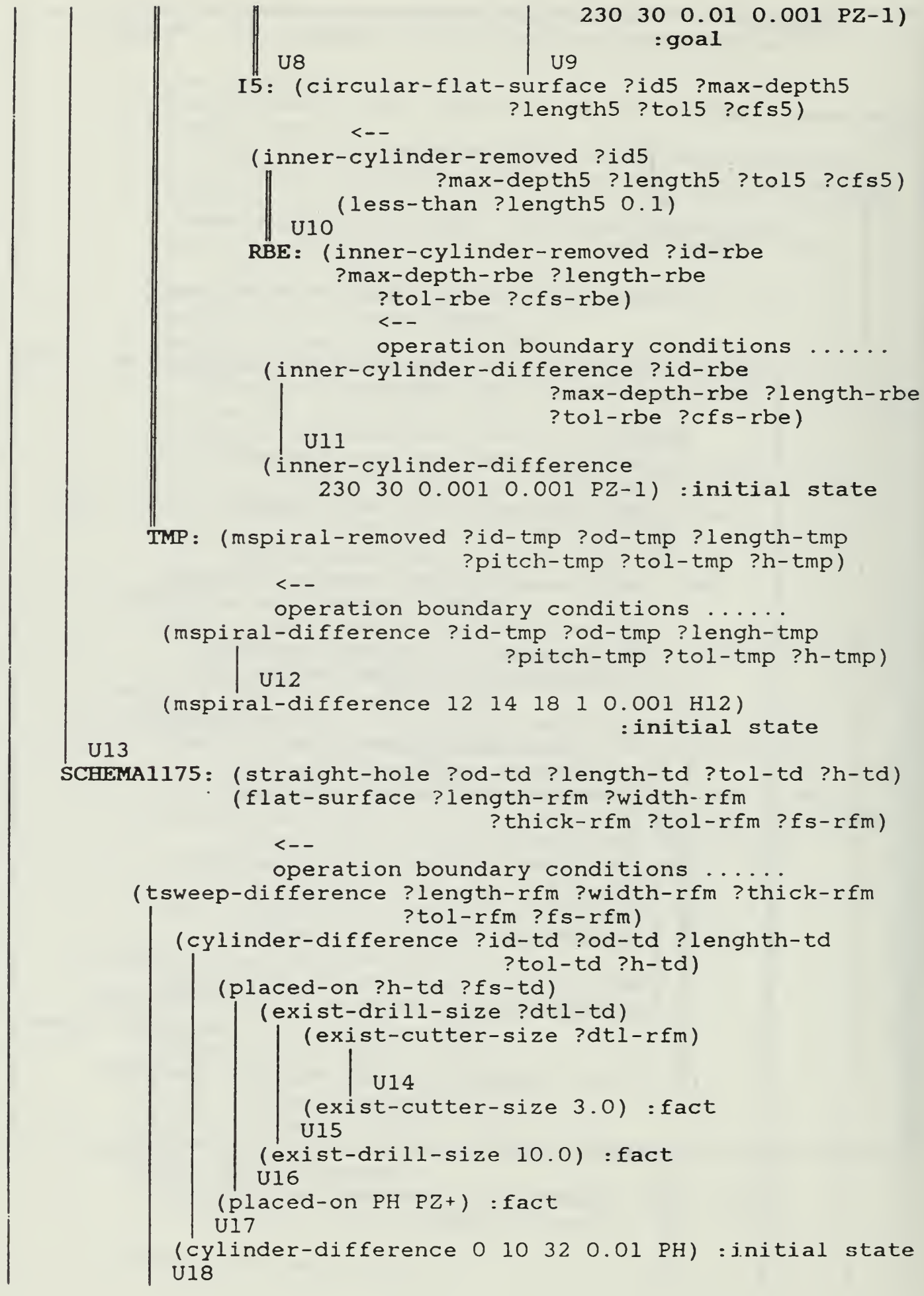
Figure 5-10 Process Plan for the Axle Cover

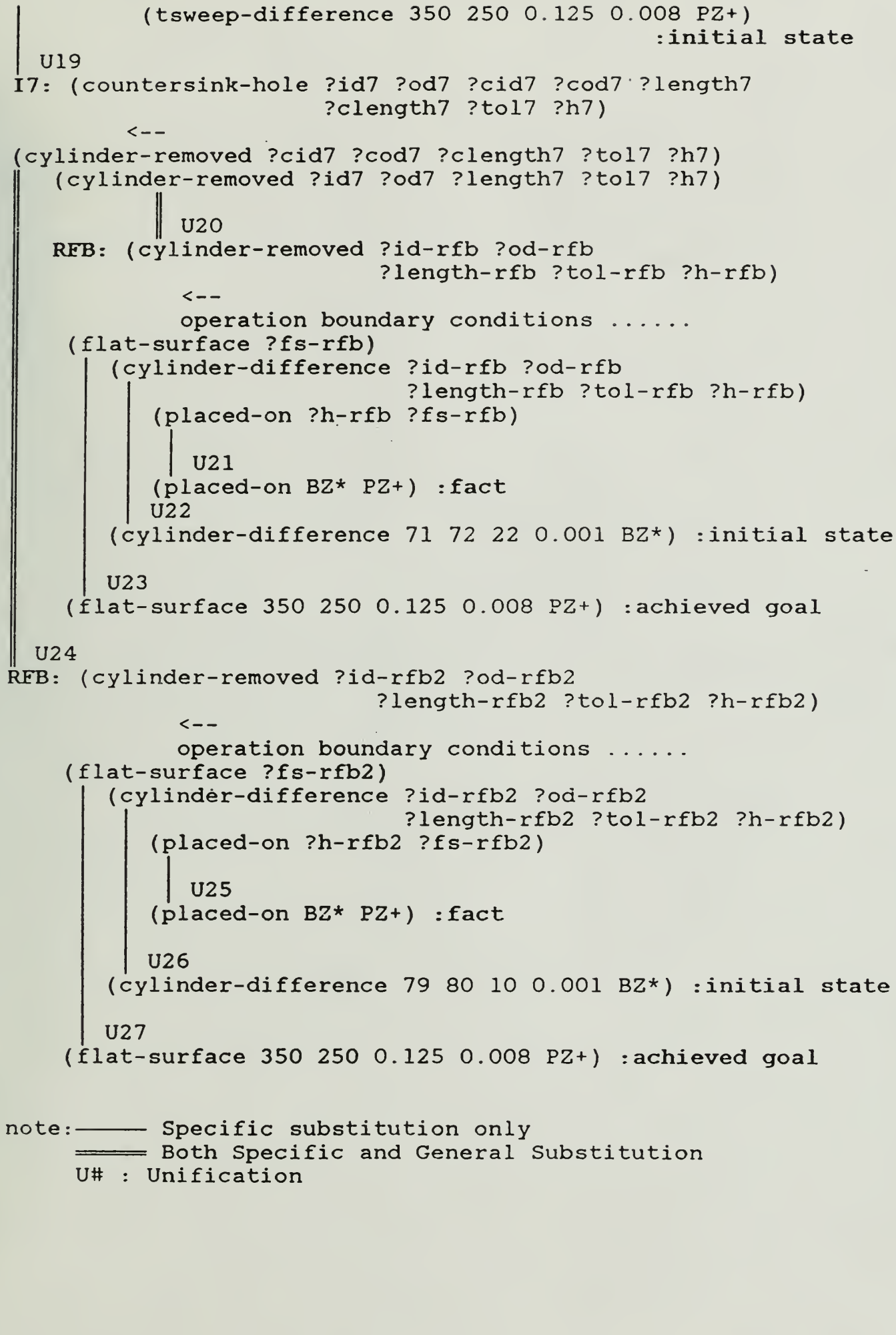
```
(SCHEMA

/* schema name */
     SCHEMA0002
/* argument list */
     (?od-tmp ?id-dmp ?length-dmp........)
/* antecedent */
     (tsweep-difference ?length-rfm ?width-rfm ?thick-rfm
          ?tol-rfm ?fs-rfm)
     (cylinder-difference ?id-td ?od-td ?lenghth-td
          ?tol-td ?h-td)
     (placed-on ?h-td ?fs-rfm)
     (exist-drill-size ?dtl-td)
     (exist-cutter-size ?dtl-rfm)
     (cylinder-difference ?id-rfb ?od-rfb ?length-rfb
          ?tol-rfb ?h-rfb)
     (placed-on ?h-rfb ?fs-rfm)
     (cylinder-difference ?id-rfb2 ?od-rfb2
          ?length-rfb2 ?tol-rfb ?h-rfb)
     (inner-cylinder-difference ?id-rbe ?max-depth-rbe
          ?length-rbe ?tol-rbe ?cfs-rbe)
     (mcylinder-difference ?id-dmp ?od-dmp
          ?length-dmp ?tol-dmp ?h-dmp)
     (placed-on ?h-dmp ?cfs-rbe)
     (exist-drill-size ?dtl-dmp)
     (mspiral-difference ?od-dmp ?od-tmp
          ?length-dmp ?pitch-tmp ?tol-dmp ?h-dmp)
     /* operation boundary conditions */
     ......
/* consequent */
     (flat-surface ?length-rfm ?width-rfm ?thick-rfm
          ?tol-rfm ?fs-?rfm)
     (straight-hole ?od-td ?lenghth-td ?tol-td ?h-td)
     (countersink-hole ?id-rfb ?od-rfb ?id-rfb2 ?od-rfb2
          ?length-rfb ?length-rfb2 ?tol-rfb ?h-rfb)
     (circular-flat-surface ?id-rbe ?max-depth-rbe
          ?length-rbe ?tol-rbe ?cfs-rbe)
     (mthreaded-hole ?id-dmp ?od-dmp ?od-tmp
          ?length-dmp ?pitch-tmp ?tol-dmp ?h-dmp)
/* generalized process plan */
     SCHEMA1175 (Rough-Finish Milling (PZ+)
                 Twist Drilling (PH)       )
     Rough-Finish Boring (BZ*)
     Rough-Finish Boring (BZ*)
     Rough Bore External (PZ-1)
     Multi Pattern Drilling(H12)
     Multi Pattern Tapping (H12)
```

Figure 5-11 Schema for Making the Axle Cover

Appendix 1. The Explanation Structure for Learning the Manufacturing Process for Making Axle Cover




230 30 0.01 0.001 PZ-1)
:goal
U8
U9
I5: (circular-flat-surface ?id5 ?max-depth5
?length5 ?tol5 ?cfs5)
<--
(inner-cylinder-removed ?id5
?max-depth5 ?length5 ?tol5 ?cfs5)
(less-than ?length5 0.1)
U10
RBE: (inner-cylinder-removed ?id-rbe
?max-depth-rbe ?length-rbe
?tol-rbe ?cfs-rbe)
<--
operation boundary conditions ......
(inner-cylinder-difference ?id-rbe
?max-depth-rbe ?length-rbe
?tol-rbe ?cfs-rbe)
U11
(inner-cylinder-difference
230 30 0.001 0.001 PZ-1) :initial state
TMP: (mspiral-removed ?id-tmp ?od-tmp ?length-tmp
?pitch-tmp ?tol-tmp ?h-tmp)
<--
operation boundary conditions ......
(mspiral-difference ?id-tmp ?od-tmp ?lengh-tmp
?pitch-tmp ?tol-tmp ?h-tmp)
U12
(mspiral-difference 12 14 18 1 0.001 H12)
:initial state
U13
SCHEMA1175: (straight-hole ?od-td ?length-td ?tol-td ?h-td)
(flat-surface ?length-rfm ?width-rfm
?thick-rfm ?tol-rfm ?fs-rfm)
<--
operation boundary conditions ......
(tsweep-difference ?length-rfm ?width-rfm ?thick-rfm
?tol-rfm ?fs-rfm)
(cylinder-difference ?id-td ?od-td ?lenghth-td
?tol-td ?h-td)
(placed-on ?h-td ?fs-td)
(exist-drill-size ?dtl-td)
(exist-cutter-size ?dtl-rfm)
U14
(exist-cutter-size 3.0) :fact
U15
(exist-drill-size 10.0) :fact
U16
(placed-on PH PZ+) :fact
U17
(cylinder-difference 0 10 32 0.01 PH) :initial state
U18



note:——— Specific substitution only
═══ Both Specific and General Substitution
U# : Unification

Process Planning System	Variant Systems	Generative Systems
Knowledge-base Contents	Library of standard process plans indexed by part family code. One part family code number identifies a group of components that bear similarities in shape, material type & processing requirement.	Process operators (e.g. Twist Drilling) indexed by their preconditions (including process boundary conditions) and effects. The Knowledge-base also contains a mechanism for recognizing geometric features such as Flat Surface, Hole, etc.
Flexibility	Capable of quickly retrieving a stored plan for components that match an existing part family code number. No stored plan for non-standard components.	Capable of generating process plans for a wide variety of components.
Response Time	A stored plan, if available, can be easily tailored to suit a given component's requirements, resulting in quick response time. Very slow response time for non-standard components which require extensive human modifications to be carried out on some standard plan.	Search for applicable operators is a function of part complexity (number of features to be created). Systems are prone to combinatorial explosion problem resulting from searching large spaces.
Existing Systems	CAPP [Link, 1976], MITURN [TNO, 1981], MIPLAN [TNO, 1981]	CPPP [Mann et al, 1977], AUTAP [Eversheim & Fuchs, 1980], TIPPS [Chang, 1984], APPAS [Wysk, 1977], GARI [Descotte & LaTombe, 1981], SIPP [Nau & Chang, 1986]

Table 1 Comparative Analysis of Process Planning Systems